


```
FFFFFFFFF 000000 RRRRRRR 000000 PPPPPPP EEEEEEEEE NN NN DDDDDDD EEEEEEEEE
FFFFFFFFF 000000 RRRRRRR 000000 PPPPPPP EEEEEEEEE NN NN DDDDDDD EEEEEEEEE
FF 00 00 RR RR 00 00 PP PP EE NN NN DD DD EE
FF 00 00 RR RR 00 00 PP PP EE NN NN DD DD EE
FF 00 00 RR RR 00 00 PP PP EE NN NN DD DD EE
FF 00 00 RR RR 00 00 PP PP EE NN NN DD DD EE
FFFFFFFFF 00 00 RRRRRRR 00 00 PPPPPPP EEEEEEEEE NN NN DD DD EEEEEEEEE
FFFFFFFFF 00 00 RRRRRRR 00 00 PPPPPPP EEEEEEEEE NN NN DD DD EEEEEEEEE
FF 00 00 RR RR 00 00 PP PP EE NN NN DD DD EE
FF 00 00 RR RR 00 00 PP PP EE NN NN DD DD EE
FF 00 00 RR RR 00 00 PP PP EE NN NN DD DD EE
FF 000000 RR RR 000000 PP PP EEEEEEEEE NN NN DDDDDDD EEEEEEEEE
FF 000000 RR RR 000000 PP PP EEEEEEEEE NN NN DDDDDDD EEEEEEEEE
```

```
LL 111111 SSSSSSS
LL 111111 SSSSSSS
LL 11 SS
LL 11 SS
LL 11 SS
LL 11 SSSSSS
LL 11 SSSSSS
LL 11 SS
LL 11 SS
LL 11 SS
LL 11 SS
LLLLLLLLL 111111 SSSSSSS
LLLLLLLLL 111111 SSSSSSS
```

```
1 0001 0 MODULE FOR$$OPEN_DEFLT (%TITLE, 'FORTRAN default open'
2 0002 0 IDENT = '1-098', ! File: FOROPENDE.B32 Edit: LEB1098
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *****
27 0027 1
28 0028 1
29 0029 1 **
30 0030 1 FACILITY: FORTRAN Support Library - not user callable
31 0031 1
32 0032 1 ABSTRACT:
33 0033 1
34 0034 1 This module contains a routine to perform default file
35 0035 1 opening for FORTRAN programs.
36 0036 1
37 0037 1 ENVIRONMENT: User access mode; mixture of AST level or not.
38 0038 1
39 0039 1 AUTHOR: Thomas N. Hastings, CREATION DATE: 6-Mar-77; Version 0
40 0040 1
41 0041 1 MODIFIED BY:
42 0042 1
43 0043 1 Thomas N. Hastings, 15-Mar-77: Version 0
44 0044 1 [Previous edit history removed. SBL 5-Oct-1982]
45 0045 1 1-078 - Add support for DEFAULTFILE=string. JAW 30-Jun-1981
46 0046 1 1-079 - Increase default value of RECL for unformatted variable-length
47 0047 1 records from 126 to 2046, to improve performance when
48 0048 1 RECORDTYPE='SEGMENTED'. JAW 17-Jul-1981
49 0049 1 1-080 - Fix logic error in record type check made when user does not
50 0050 1 specify record type for an old file. (Allowed both FIXED and
51 0051 1 SEGMENTED to be set simultaneously.) JAW 25-Aug-1981
52 0052 1 1-081 - Change algorithm for determining the length of a list-directed
53 0053 1 output record: use RECL if specified, else 80/81 depending on
54 0054 1 carriage control. JAW 26-Aug-1981
55 0055 1 1-082 - Add test for blocksize less than recordsize (made only if open
56 0056 1 or create fails and device is mag tape). If so, signal
57 0057 1 INCRELEN since RMS does not give a useful message in this
```

```
.. 58      0058 1 | case. JAW 28-Aug-1981
.. 59      0059 1 | 1-083 - Save and restore the STS and STV around the $PARSE we do if we
.. 60      0060 1 | get an unexpected error. SBL 28-Sep-1981
.. 61      0061 1 | 1-084 - Signal FOR$K_OPEFAI if RMS$ WLK and not readonly. DGP 03-Dec-1981
.. 62      0062 1 | 1-085 - Set the MRS in the FAB for indexed files. DGP 21-Dec-1981
.. 63      0063 1 | 1-086 - Allow existing file to be SEGMENTED only if it has RFM=VAR.
.. 64      0064 1 | Correct 1-082 and 1-084 so that only RMS$_CRE errors check for
.. 65      0065 1 | INCRECLEN. SBL 13-Jan-1982
.. 66      0066 1 | 1-087 - Complete 1-085. It was much too simplistic and caused existing ISAM
.. 67      0067 1 | files to not be able to opened. DGP 22-Feb-1982
.. 68      0068 1 | 1-088 - Unfortunately, 1-087 did not allow existing ISAM files with an MRS
.. 69      0069 1 | smaller than the default buffer size to be opened unless the
.. 70      0070 1 | RECL was explicitly specified. Fix it. SBL 16-Apr-1982
.. 71      0071 1 | 1-089 - For devices other than disks and terminals, reduce the default
.. 72      0072 1 | recordsize to less than the blocksize, if necessary. Use blocksize
.. 73      0073 1 | as recordsize on existing files, if no MRS or LRL. SBL 30-Sep-1982
.. 74      0074 1 | 1-090 - Make default unformatted RECL 2044 instead of 2046. This allows
.. 75      0075 1 | default disk files to be copied to tape. SBL 8-Nov-1982
.. 76      0076 1 | 1-091 - Reflect change that OT$$$ data structures are now FOR$$$. SBL 8-Nov-1982
.. 77      0077 1 | 1-092 - Restore some INCOPECLO checks that were mistakenly deleted
.. 78      0078 1 | in an earlier edit. Use new macro to call FOR$$$SIGNAL_STO.
.. 79      0079 1 | Move FAB and NAM to heap at end of RAB. Add support for stream
.. 80      0080 1 | recordtypes. Raise bucketsize limit to 63. Use carriagecontrol
.. 81      0081 1 | specified/defaulted if PPF. Don't decrement recordlength by 4
.. 82      0082 1 | unless SEGMENTED. SBL 29-Mar-1983
.. 83      0083 1 | 1-093 - Add RFA cacheing for BACKSPACE. SBL 2-June-1983
.. 84      0084 1 | 1-094 - Restrict stream recordtypes to sequential org only. SBL 28-Jul-1983
.. 85      0085 1 | 1-095 - Use LNM$C_NAMLENGTH for maximum size of equivalence string in call
.. 86      0086 1 | to $TRNLOG. DG 8-Nov-1983
.. 87      0087 1 | 1-096 - Add stack location of TEMP_FNS to store the temporary filespec
.. 88      0088 1 | for ASSIGN. Also change back use of LNM$C_NAMLENGTH to be
.. 89      0089 1 | NAM$C_MAXRSS. LEB 2-Feb-1984
.. 90      0090 1 | 1-097 - Free KEY_XABs when an open fails. STAN 27-Feb-1984.
.. 91      0091 1 | 1-098 - Disassociate the NAM block during the $PARSE to clear up a
.. 92      0092 1 | problem associated with floating memory. LEB 21-Mar-1984
.. 93      0093 1 | --
.. 94      0094 1 |
```

```

96      0095 1 |
97      0096 1 | PROLOGUE FILE:
98      0097 1 |
99      0098 1 |
100     0099 1 | REQUIRE 'RTLIN:FORPROLOG';           ! FORTRAN definitions
101     0165 1 |
102     0166 1 |
103     0167 1 | TABLE OF CONTENTS:
104     0168 1 |
105     0169 1 |
106     0170 1 | FORWARD ROUTINE
107     0171 1 |     FOR$OPEN_DEFLT : CALL_CCB NOVALUE,   ! default OPEN
108     0172 1 |     FOR$OPEN_PROC : CALL_CCB NOVALUE;    ! common OPEN procedure
109     0173 1 |
110     0174 1 |
111     0175 1 | MACROS:
112     0176 1 |
113     0177 1 |     NONE
114     0178 1 |
115     0179 1 | EQUATED SYMBOLS:
116     0180 1 |
117     0181 1 |     NONE
118     0182 1 |
119     0183 1 | OWN STORAGE:
120     0184 1 |
121     0185 1 |     NONE
122     0186 1 |
123     0187 1 | EXTERNAL REFERENCES:
124     0188 1 |
125     0189 1 |
126     0190 1 | EXTERNAL ROUTINE
127     0191 1 |     FOR$ERR_OPECLO,
128     0192 1 |     FOR$$$SIG_STO : NOVALUE,           ! OPEN/CLOSE condition handler
129     0193 1 |                                           ! Convert small FORTRAN err #
130     0194 1 |     FOR$$$SIG_NO_LUB : NOVALUE,        ! to 32-bit VAX error # and SIGNAL_STOP
131     0195 1 |                                           ! same as FOR$$$SIG_STO except no LUB setup
132     0196 1 |     FOR$CB_PUSH : JSB_CB_PUSH NOVALUE, ! so must pass LUN explicitly.
133     0197 1 |                                           ! push current LUB/ISB/RAB, if any, and allocate LUB/ISB/RAB
134     0198 1 |     FOR$CB_POP : JSB_CB_POP NOVALUE,   ! for this logical unit
135     0199 1 |                                           ! Pop I/O system back to previous LUB or indicate
136     0200 1 |     FOR$GET_VM,                        ! no I/O statement is currently being processed.
137     0201 1 |     FOR$FREE_VM : NOVALUE,             ! Allocate virtual memory
138     0202 1 |     FOR$$$SIG_FATINT : NOVALUE,        ! Free virtual memory
139     0203 1 |     FOR$DECL_EXITH : NOVALUE;          ! Signal_stop internal error
140     0204 1 |                                           ! Declare the exit handler
141     0205 1 | EXTERNAL
142     0206 1 |     FOR$XL_XIT_LOCK;                   ! True if exit handler already declared
143     0207 1 |
```

```
145 0208 1 GLOBAL ROUTINE FOR$OPEN_DEFLT (      ! Default OPEN
146 0209 1     ACCESS_VAL,                      ! Access = OPEN$K_ACC {SEQ, DIR}
147 0210 1     TYPE_VAL,                        ! TYPE = OPEN$K_ACC {NEW, OLD}
148 0211 1     FORM_VAL,                       ! FORM = OPEN$K_FOR {UNF, FOR, UNS}
149 0212 1     : CALL_CCB NOVALUE =
150 0213 1
151 0214 1 ++
152 0215 1 ABSTRACT:
153 0216 1
154 0217 1     Perform default OPEN for an I/O statement for the indicated
155 0218 1     logical unit. The possible parameters are a restricted
156 0219 1     subset of explicit OPEN, plus FORM = 'UNSPECIFIED' (for
157 0220 1     ENDFILE only). The keywords for default OPEN are:
158 0221 1     ACCESS, TYPE, and FORM.
159 0222 1
160 0223 1 FORMAL PARAMETERS:
161 0224 1
162 0225 1     LUB_ADR.mlu.ra      adr of LUB/ISB/RAB control block
163 0226 1     ACCESS_VAL.rlu.v   Value = OPEN$K_ACC {SEQ, DIR}
164 0227 1                   to indicate ACCESS = 'SEQUENTIAL'
165 0228 1                   or 'DIRECT'.
166 0229 1     TYPE_VAL.rlu.v      Value = OPEN$K_TYPE {NEW, OLD} TO
167 0230 1                   indicate TYPE = 'NEW' or 'OLD'
168 0231 1     FORM_VAL.rlu.v      Value = OPEN$K_FORM {UNF, FOR, UNS}
169 0232 1                   to indicate FORM = 'UNFORMATTED',
170 0233 1                   'FORMATTED', or 'UNSPECIFIED'
171 0234 1                   (ENDFILE only).
172 0235 1
173 0236 1 IMPLICIT INPUTS:
174 0237 1
175 0238 1     LUB$V_READ_ONLY    1 if 'READONLY' specified in CALL FDBSET
176 0239 1     LUB$V_DIRECT       1 if specified on previous DEFINEFILE
177 0240 1     LUB$V_OLD_FILE     1 if specified on previous CALL FDBSET
178 0241 1     LUB$V_UNFORMAT     1 if specified on previous DEFINEFILE
179 0242 1     LUB$W_LUN          FORTRAN logical unit number
180 0243 1
181 0244 1 IMPLICIT OUTPUTS:
182 0245 1
183 0246 1     LUB$V_DIRECT       1 if ACCESS = 'DIRECT' or DEFINEFILE
184 0247 1     LUB$V_OLD_FILE     1 if TYPE = 'OLD' or CALL FDBSET 'OLD'
185 0248 1     LUB$V_FORMATTED    1 if FORM = 'FORMATTED'
186 0249 1     LUB$V_UNFORMAT     1 if FORM = 'UNFORMATTED' or DEFINEFILE
187 0250 1
188 0251 1 COMPLETION STATUS:
189 0252 1
190 0253 1     NONE
191 0254 1
192 0255 1 SIDE EFFECTS:
193 0256 1
194 0257 1     See FOR$OPEN_PROC for SIGNAL_STOPS.
195 0258 1 --
196 0259 1
197 0260 2 BEGIN
198 0261 2
199 0262 2 EXTERNAL REGISTER
200 0263 2     CCB : REF $FOR$CCB_DECL;
201 0264 2
```

```
202 0265 2 LOCAL
203 0266 OPEN : VECTOR [OPEN$K_KEY_MAX + 1]; ! OPEN parameter array
204 0267
205 0268
206 0269
207 0270
208 0271
209 0272 CH$FILL (0, (OPEN$K_KEY_MAX + 1)*%UPVAL, OPEN);
210 0273
211 0274
212 0275
213 0276
214 0277
215 0278 OPEN [OPEN$K_ACCESS] = .ACCESS_VAL;
216 0279 OPEN [OPEN$K_TYPE] = .TYPE_VAL;
217 0280 OPEN [OPEN$K_FORM] = .FORM_VAL;
218 0281
219 0282
220 0283
221 0284
222 0285
223 0286
224 0287
225 0288
226 0289 1
```

! End of FOR\$OPEN_DEFLT routine

.TITLE FOR\$OPEN_DEFLT FORTRAN default open
.IDENT \1-098\

.EXTRN FOR\$ERR OPECLO
.EXTRN FOR\$SIGAL_STO
.EXTRN FOR\$SIG NO_LUB
.EXTRN FOR\$CB_PUSH, FOR\$CB_POP
.EXTRN FOR\$GET_VM, FOR\$FREE_VM
.EXTRN FOR\$SIG_FATINT
.EXTRN FOR\$DECC_EXITH
.EXTRN FOR\$SL_XIT_LOCK

.PSECT _FOR\$CODE, NOWRT, SHR, PIC, 2

.ENTRY FOR\$OPEN_DEFLT, Save R2,R3,R4,R5
MOVAB -108(SP), SP
MOVCS #0, (SP), #0, #108, OPEN
MOVL ACCESS_VAL, OPEN+16
MOVL TYPE_VAL, OPEN+60
MOVL FORM_VAL, OPEN+20
PUSHL SP
CALLS #1, FOR\$OPEN_PROC
RET

: 0208
: 0272
: 0278
: 0279
: 0280
: 0287
: 0289

```
006C 8F 00 5E 94 AE 9E 00002
6E 00 2C 00006
6E 0000D
10 AE 04 AC D0 0000E
3C AE 08 AC D0 00013
14 AE 0C AC D0 00018
0000V CF 01 FB 0001F
04 00024
```

; Routine Size: 37 bytes, Routine Base: _FOR\$CODE + 0000

; 227 0290 1

```
229 0291 1 GLOBAL ROUTINE FOR$OPEN_PROC (      ! Do an OPEN
230 0292 1   OPEN_ADR)                        ! Address of OPEN parameter vector
231 0293 1   : CALL_CCB NOVALUE =
232 0294 1
233 0295 1 ++
234 0296 1 ABSTRACT:
235 0297 1
236 0298 1   This routine performs the OPEN for FOR$OPEN and FOR$OPEN_DEFLT.
237 0299 1   The OPEN parameters have been picked up and placed in a
238 0300 1   longword array. The index is parameter specific. The parameters
239 0301 1   are processed in a logical order which minimizes the
240 0302 1   distance between parameters which depend on each other.
241 0303 1   Each parameter sets an appropriate part of the LUB/ISB/RAB
242 0304 1   control block or the FAB control block. If the FAB
243 0305 1   has not been allocated, it is allocated.
244 0306 1   Whenever the FAB, RAB, LUB, or ISB
245 0307 1   are allocated they are initially set to 0. Thus, default values
246 0308 1   are often indicated by zero in these structures.
247 0309 1
248 0310 1 FORMAL PARAMETERS:
249 0311 1
250 0312 1   LUB_ADR.mlu.ra      Adr. of LUB/ISB/RAB control block
251 0313 1   OPEN_ADR.mlu.ra  Adr. of OPEN parameter array of
252 0314 1                     longwords. Index is of form:
253 0315 1   OPEN$K_name. A longword value of 0
254 0316 1                     indicates an omitted keyword.
255 0317 1
256 0318 1 IMPLICIT INPUTS:
257 0319 1
258 0320 1   LUB$V_READ_ONLY      1 if 'READONLY' specified in CALL FDBSET
259 0321 1   LUB$V_DIRECT        1 if specified on previous DEFINEFILE
260 0322 1   LUB$V_OLD_FILE      1 if specified on previous CALL FDBSET
261 0323 1   LUB$V_UNFORMAT      1 if specified on previous DEFINEFILE
262 0324 1   LUB$W_LUN          FORTRAN logical unit number
263 0325 1   LUB$W_RBUF_SIZE   Size in bytes of record buffer to be allocated
264 0326 1
265 0327 1 IMPLICIT OUTPUTS:
266 0328 1
267 0329 1   LUB$V_READ_ONLY      1 if 'READONLY' present or CALL FDBSET
268 0330 1   LUB$V_DIRECT        1 if ACCESS = 'DIRECT' or DEFINEFILE
269 0331 1   LUB$V_OLD_FILE      1 if TYPE = 'OLD' or CALL FDBSET 'OLD'
270 0332 1   LUB$V_SCRATCH       1 if TYPE = 'SCRATCH'
271 0333 1   LUB$V_PRINT          1 if DISPOSE = 'PRINT'
272 0334 1   LUB$V_FIXED         1 if RECORDTYPE = 'FIXED'
273 0335 1   LUB$V_FORMATTED      1 if FORM = 'FORMATTED' or omitted
274 0336 1   LUB$V_UNFORMAT      1 if FORM = 'UNFORMATTED'
275 0337 1                     or DEFINEFILE
276 0338 1   LUB$A_ASSOC_VAR      adr. of n if ASSOCIATEVARIABLE = n is present
277 0339 1                     in OPEN or DEFINEFILE
278 0340 1   LUB$V_ASS_VAR_L      1 if n is longword
279 0341 1   LUB$W_IFI           RMS internal file id. Needed in case
280 0342 1                     FORTRAN CLOSE done.
281 0343 1   LUB$W_RBUF_SIZE      Size in bytes of record buffer allocated.
282 0344 1   LUB$L_LOG_RECNO      1
283 0345 1   LUB$W_R_MARGIN      List directed output line width
284 0346 1   LUB$B_ORGAN          Organization, either LUB$K_ORG_SEQUE
285 0347 1                     or LUB$K_ORG_RELAT.
```

```
286 0348 1 |
287 0349 1 | COMPLETION STATUS:
288 0350 1 |
289 0351 1 |     NONE
290 0352 1 |
291 0353 1 | SIDE EFFECTS:
292 0354 1 |
293 0355 1 |     SIGNAL STOPS the following errors:
294 0356 1 |     FOR$_FILNOTFOU (29 = 'FILE NOT FOUND')
295 0357 1 |     FOR$_OPEFAI (30 = 'OPEN FAILURE')
296 0358 1 |     FOR$_INCRECLEN (37 = 'INCONSISTENT RECORD LENGTH')
297 0359 1 |     FOR$_INSVIRMEM (41 = 'INSUFFICIENT VIRTUAL MEMORY')
298 0360 1 |     FOR$_NO_SUCDEV (42 = 'NO SUCH DEVICE')
299 0361 1 |     FOR$_FICNAMSPE (43 = 'FILE NAME SPECIFICATION ERROR')
300 0362 1 |     FOR$_RECSPEERR (44 = 'RECORD SPECIFICATION ERROR')
301 0363 1 |     FOR$_KEYVALERR (45 = 'KEYWORD VALUE ERROR IN OPEN STATEMENT')
302 0364 1 |     FOR$_INCOPECLO (46 = 'INCONSISTENT OPEN/CLOSE ARGUMENTS')
303 0365 1 |     FOR$_INVARGFOR (47 = 'INVALID ARGUMENT TO FORTRAN I/O LIBRARY')
304 0366 1 | --
305 0367 1 |
306 0368 2 | BEGIN
307 0369 2 |
308 0370 2 | EXTERNAL REGISTER
309 0371 2 |     CCB : REF $FOR$CCB_DECL;
310 0372 2 |
311 0373 2 | MAP
312 0374 2 |     OPEN_ADR : REF VECTOR [OPEN$K_KEY_MAX + 1];
313 0375 2 |
314 0376 2 | LOCAL
315 0377 2 |     V DEFAULT SIZE,
316 0378 2 |     OPEN_STATOS,
317 0379 2 |     T_DFCT_FILE_NAM : VECTOR [10, BYTE],
318 0380 2 |
319 0381 2 |     ORIG_RAT: BYTE,
320 0382 2 |     XAB_BLOCK : BLOCK [XAB$C_FHCLEN, BYTE],
321 0383 2 |     KEY_XAB : REF BLOCK [OPEN$K_XAB_SIZE, BYTE],
322 0384 2 |     TEMP_FNS: VECTOR [NAM$C_MAXRSS, BYTE],
323 0385 2 |     RES_OR_EXP_NAME : VECTOR [NAM$C_MAXRSS, BYTE];
324 0386 2 |
325 0387 2 | BIND
326 0388 2 |     FAB = CCB: REF $FOR$FAB_CCB_STRUCT,
327 0389 2 |     NAM = CCB: REF $FOR$NAM_CCB_STRUCT,
328 0390 2 |     A_SYSS$INPUT = UPLIT BYTE('SYSS$INPUT:'),
329 0391 2 |     A_SYSS$OUTPUT = UPLIT BYTE('SYSS$OUTPUT:');
330 0392 2 |
331 0393 2 | BUILTIN
332 0394 2 |     TESTBITSC;
333 0395 2 |
334 0396 2 | LITERAL
335 0397 2 |     L_SYSS$INPUT = %CHARCOUNT ('SYSS$INPUT:'),
336 0398 2 |     L_SYSS$OUTPUT = %CHARCOUNT ('SYSS$OUTPUT:');
337 0399 2 |
338 0400 2 |
339 0401 2 |     !+ See if ASSIGN or FDBSET has already allocated us a FAB. If so,
340 0402 2 |     ! copy it to our local FAB and deallocate it. Copy the filename too
341 0403 2 |     ! if it's there.
342 0404 2 |
```

```
343 0405 2
344 0406 2
345 0407 2
346 0408 2
347 0409 2
348 0410 2
349 0411 2
350 0412 2
351 0413 2
352 0414 2
353 0415 2
354 0416 2
355 0417 2
356 0418 2
357 0419 2
358 0420 2
359 0421 2
360 0422 2
361 0423 2
362 0424 2
363 0425 2
364 0426 2
365 0427 2
366 0428 2
367 0429 2
368 0430 2
369 0431 2
370 0432 2
371 0433 2
372 0434 2
373 0435 2
374 0436 2
375 0437 2
376 0438 2
377 0439 2
378 0440 2

IF .CCB [LUB$A_FAB] NEQA 0
THEN
  BEGIN
  LOCAL
    HEAP_FAB: REF BLOCK [, BYTE];
    HEAP_FAB = .CCB [LUB$A_FAB];
    CHSMOVE (.HEAP_FAB [FAB$B_BLN], .HEAP_FAB, FAB [0,0,0,0]);
    FOR$FREE_VM (.HEAP_FAB [FAB$B_BLN], .HEAP_FAB);
    CCB [LUB$A_FAB] = 0;
    IF .FAB [FAB$B_FNS] NEQU 0
    THEN
      BEGIN
        CHSMOVE (.FAB [FAB$B_FNS], .FAB [FAB$L_FNA], TEMP_FNS);
        FOR$FREE_VM (.FAB [FAB$B_FNS], .FAB [FAB$L_FNA]);
        FAB [FAB$L_FNA] = TEMP_FNS;
      END;
    END;

  !+
  !- Initialize NAM and FHC XAB_BLOCK.
  !-
  FAB [FAB$L_NAM] = NAM [0,0,0,0];
  NAM [NAM$L_RSA] = NAM [NAM$L_ESA] = RES OR EXP NAME;
  NAM [NAM$B_RSS] = NAM [NAM$B_ESS] = NAM$C_MAXRSS;
  $XABFHC INIT (XAB = XAB_BLOCK);
  FAB [FAB$L_XAB] = XAB_BLOCK;
  KEY_XAB = XAB_BLOCK; ! First XAB in chain

  !+
  !- Set deferred write bit in the FAB for speed improvement in
  !- relative files.
  !-
  FAB [FAB$V_DFW] = 1;
```

```
380 0441 2 !
381 0442
382 0443
383 0444
384 0445
385 0446
386 0447
387 0448
388 0449
389 0450
390 0451
391 0452
392 0453
393 0454
394 0455
395 0456
396 0457
397 0458
398 0459
399 0460
400 0461
401 0462
402 0463
403 0464
404 0465
405 0466
406 0467
407 0468
408 0469
409 0470
410 0471
411 0472
412 0473
413 0474
414 0475
415 0476
416 0477
417 0478
418 0479
419 0480
420 0481
421 0482
422 0483
423 0484
424 0485
425 0486
426 0487
427 0488
428 0489
429 0490
430 0491
431 0492
432 0493
433 0494
434 0495
435 0496
436 0497
```

```
!
NAME
Setup RMS default filename string (FAB$L_DNA, FAB$B_DNS) and
file name string (FAB$F_FNA) depending on the type of statement
that caused the LUN to be opened.

      statement      file name string      default file name string

      READ           FOR$READ:           FORREAD.DAT
      ACCEPT         FOR$ACCEPT:         FORACCEPT.DAT
      TYPE           FOR$TYPE:           FORTYPE.DAT
      PRINT          FOR$PRINT:          FORPRINT.DAT
      other          FORnnn:             FORnnn.DAT

Get the logical unit number from LUB$W_LUN instead of
OPEN[OPEN$K_UNIT] since default open doesn't set up UNIT.
LUN has been checked for being in legal range by CB_PUSH.
Set the string length and address in the FAB.

BEGIN

LOCAL
  A_DEF_LOGNAM,      ! Address of default logical name
  L_DEF_LOGNAM;      ! Length of default logical name

A_DEF_LOGNAM = 0;    ! No default yet

CASE .CCB [LUB$W_LUN] FROM LUB$K_DLUN_MIN TO LUB$K_DLUN_MAX OF
  SET
    [LUB$K_LUN_READ] :      ! READ statement (therefore default open)
      BEGIN
        FAB [FAB$B_DNS] = %CHARCOUNT ('FORREAD.DAT');
        FAB [FAB$F_DNA] = UPLIT BYTE('FORREAD.DAT');
        FAB [FAB$B_FNS] = %CHARCOUNT ('FOR$READ:');
        FAB [FAB$F_FNA] = UPLIT BYTE('FOR$READ:');
        A_DEF_LOGNAM = A_SYSSINPUT;
        L_DEF_LOGNAM = L_SYSSINPUT;
      ERD;

    [LUB$K_LUN_ACCE] :      ! ACCEPT statement (therefore default open)
      BEGIN
        FAB [FAB$B_DNS] = %CHARCOUNT ('FORACCEPT.DAT');
        FAB [FAB$F_DNA] = UPLIT BYTE('FORACCEPT.DAT');
        FAB [FAB$B_FNS] = %CHARCOUNT ('FOR$ACCEPT:');
        FAB [FAB$F_FNA] = UPLIT BYTE('FOR$ACCEPT:');
        A_DEF_LOGNAM = A_SYSSINPUT;
        L_DEF_LOGNAM = L_SYSSINPUT;
      ERD;

    [LUB$K_LUN_TYPE] :      ! TYPE statement (therefore default open)
      BEGIN
        FAB [FAB$B_DNS] = %CHARCOUNT ('FORTYPE.DAT');
        FAB [FAB$F_DNA] = UPLIT BYTE('FORTYPE.DAT');
```

```
437      FAB [FAB$B_FNS] = %CHARCOUNT ('FOR$TYPE:');
438      FAB [FAB$L_FNA] = UPLIT BYTE('FOR$TYPE:');
439      A_DEF_LOGNAM = A_SYSS$OUTPUT;
440      L_DEF_LOGNAM = L_SYSS$OUTPUT;
441      END;
442
443      [LUB$K_LUN_PRIN] :                                ! PRINT statement (therefore default open)
444      BEGIN
445      FAB [FAB$B_DNS] = %CHARCOUNT ('FORPRINT.DAT');
446      FAB [FAB$L_DNA] = UPLIT BYTE('FORPRINT.DAT');
447      FAB [FAB$B_FNS] = %CHARCOUNT ('FOR$PRINT:');
448      FAB [FAB$L_FNA] = UPLIT BYTE('FOR$PRINT:');
449      A_DEF_LOGNAM = A_SYSS$OUTPUT;
450      L_DEF_LOGNAM = L_SYSS$OUTPUT;
451      END;
452
453      [OUTRANGE] :                                       ! Some other statement (OPEN or default OPEN)
454      BEGIN
455      IF .OPEN_ADR [OPENS$K_NAME] EQLA 0 OR
456      .OPEN_ADR [OPENS$K_DEFAULTF] EQLA 0
457      THEN
458      BEGIN
459      T_DFLT_FILE_NAM [0] = %C'F';
460      T_DFLT_FILE_NAM [1] = %C'D';
461      T_DFLT_FILE_NAM [2] = %C'R';
462      T_DFLT_FILE_NAM [3] = ((.CCB [LUB$W_LUN]/100) MOD 10) + %C'0';
463      T_DFLT_FILE_NAM [4] = ((.CCB [LUB$W_LUN]/10) MOD 10) + %C'0';
464      T_DFLT_FILE_NAM [5] = ((.CCB [LUB$W_LUN]) MOD 10) + %C'0';
465      T_DFLT_FILE_NAM [6] = %C'.';
466      T_DFLT_FILE_NAM [7] = %C'D';
467      T_DFLT_FILE_NAM [8] = %C'A';
468      T_DFLT_FILE_NAM [9] = %C'T';
469      END;
470
471      !+
472      ! DEFAULTFILE
473      ! Set up default file name string to be used in RMS $OPEN
474      !-
475
476      IF .OPEN_ADR [OPENS$K_DEFAULTF] NEQA 0
477      THEN
478      BEGIN
479      LOCAL
480      NAM_DSC : REF BLOCK [8, BYTE];
481      NAM_DSC = .OPEN_ADR [OPENS$K_DEFAULTF];
482      IF .NAM_DSC [DSC$W_LENGTH] GTRU 255 THEN $FOR$$SIGNAL_STO (FOR$K_FILNAMSPE);
483      FAB [FAB$B_DNS] = .NAM_DSC [DSC$W_LENGTH];
484      FAB [FAB$L_DNA] = .NAM_DSC [DSC$A_POINTER];
485      END
486      ELSE
487
488      !+
489      ! Default file name not specified in OPEN or this is default OPEN.
490      !-
491
492      BEGIN
493      FAB [FAB$B_DNS] = %CHARCOUNT ('FORnnn.DAT');
```

```
494      0555  5      FAB [FAB$L_DNA] = T_DFLT_FILE_NAM;  
495      0556  4      END;  
496      0557  4  
497      0558  4  
498      0559  4  
499      0560  4  
500      0561  4  
501      0562  4  
502      0563  4  
503      0564  4  
504      0565  4  
505      0566  4  
506      0567  4  
507      0568  4  
508      0569  4  
509      0570  4  
510      0571  4  
511      0572  4  
512      0573  4  
513      0574  4  
514      0575  4  
515      0576  4  
516      0577  4  
517      0578  4  
518      0579  4  
519      0580  4  
520      0581  4  
521      0582  4  
522      0583  4  
523      0584  4  
524      0585  4  
525      0586  4  
526      0587  4  
527      0588  4  
528      0589  4  
529      0590  4  
530      0591  4  
531      0592  4  
532      0593  4  
533      0594  4  
534      0595  4  
535      0596  4  
536      0597  4  
537      0598  4  
538      0599  4  
539      0600  4  
540      0601  4  
541      0602  4  
542      0603  4  
543      0604  4  
544      0605  4  
545      0606  4  
546      0607  4  
547      0608  4  
548      0609  4  
549      0610  4  
550      0611  4  
  
      FILE  
      Setup file name string to be used in RMS $OPEN  
  
      IF .OPEN_ADR [OPENS$K_NAME] NEQA 0  
      THEN  
      BEGIN  
          file name specified in OPEN  
          Set length and address in FAB  
  
          LOCAL  
            NAM_DSC : REF BLOCK [8, BYTE];      ! File name descriptor  
          NAM_DSC = .OPEN_ADR [OPENS$K_NAME];      ! Get descriptor  
          IF .NAM_DSC [DSC$W_LENGTH] GTRU 255 THEN $FOR$$SIGNAL_STO (FOR$K_FILNAMSPE);  
          FAB [FAB$B_FNS] = .NAM_DSC [DSC$W_LENGTH];  
          FAB [FAB$L_FNA] = .NAM_DSC [DSC$A_POINTER];  
      END  
      ELSE  
          File name not specified in OPEN or this is default OPEN.  
          If name not already setup (CALL ASSIGN), use all but last 4 characters of default filename str  
          i.e., all characters but .DAT  
          Thus filename string is a string with no punctuation so it can be a logical name  
  
          IF .FAB [FAB$L_FNA] EQLA 0  
          THEN  
          BEGIN  
              FAB [FAB$B_FNS] = %CHARCOUNT ('FORnnn');  
              FAB [FAB$L_FNA] = T_DFLT_FILE_NAM;  
  
              If this is unit 5 or 6, set up default logical  
              name to use if translation of FOR005 or FOR006  
              fails.  
  
              IF .CCB [LUB$W_LUN] EQL 5  
              THEN  
              BEGIN  
                  A_DEF_LOGNAM = A_SYSS$INPUT;  
                  L_DEF_LOGNAM = L_SYSS$INPUT;  
              END  
          ELSE
```

```
551 0612 5 IF .CCB [LUB$W_LUN] EQL 6
552 0613 5 THEN
553 0614 6 BEGIN
554 0615 6 A_DEF_LOGNAM = A_SYSS$OUTPUT;
555 0616 6 L_DEF_LOGNAM = L_SYSS$OUTPUT;
556 0617 5 END;
557 0618 5
558 0619 5 END;
559 0620 5
560 0621 5 END; ! End OUTRANGE expression
561 0622 5
562 0623 5
563 0624 5
564 0625 5 !+
565 0626 5 If we have an implicit logical name assignment possible
566 0627 5 (unit<0 or unit=5 or unit=6) then attempt translation of
567 0628 5 the logical name. If it fails, then substitute the default
568 0629 5 logical name SYSS$INPUT: or SYSS$OUTPUT: appropriately.
569 0630 5
570 0631 5 IF .A_DEF_LOGNAM NEQ 0
571 0632 5 THEN
572 0633 5 BEGIN
573 0634 5
574 0635 5 LOCAL
575 0636 5 LOGNAM_DSC : DSC$DESCRIPTOR, ! Logical name descriptor
576 0637 5 RESULT_DSC : DSC$DESCRIPTOR; ! Translation result descriptor
577 0638 5
578 0639 5 LOGNAM_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
579 0640 5 LOGNAM_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
580 0641 5 RESULT_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
581 0642 5 RESULT_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
582 0643 5 RESULT_DSC [DSC$W_LENGTH] = NAM$C MAXRSS; ! Scratch string
583 0644 5 RESULT_DSC [DSC$A_POINTER] = RES OR EXP NAME;
584 0645 5 LOGNAM_DSC [DSC$A_POINTER] = .FAB [FAB$C_FNA];
585 0646 5 LOGNAM_DSC [DSC$W_LENGTH] = .FAB [FAB$B_FNS];
586 0647 5
587 0648 5 IF .CCB [LUB$W_LUN] LSS 0
588 0649 5 THEN
589 0650 5
590 0651 5 !+
591 0652 5 Don't translate trailing colon.
592 0653 5
593 0654 5
594 0655 5 LOGNAM_DSC [DSC$W_LENGTH] = .LOGNAM_DSC [DSC$W_LENGTH] - 1;
595 0656 5
596 0657 5 !+
597 0658 5 Attempt to translate the logical name, putting the result in
598 0659 5 RES_OR_EXP NAME. We don't care what it translated to, just
599 0660 5 the fact that it does translate. If it does not, then substitute
600 0661 5 the default logical name for the file name.
601 0662 5
602 0663 5
603 0664 5 IF $TRNLOG (LOGNAM = LOGNAM_DSC, RSLBUF = RESULT_DSC) EQLU SSS_NOTRAN
604 0665 5 THEN
605 0666 5 BEGIN
606 0667 5 FAB [FAB$C_FNA] = .A_DEF_LOGNAM;
607 0668 5 FAB [FAB$B_FNS] = .L_DEF_LOGNAM;
```

```

: 608      0669      4      END;
: 609      0670      4
: 610      0671      4      END;
: 611      0672      4
: 612      0673      4      END;
: 613      0674      4
: 614      0675      4      !+
: 615      0676      4      !- Set the filename in the LUB in case an error occurs.
: 616      0677      4
: 617      0678      4
: 618      0679      4      CCB [LUB$A_RSN] = .FAB [FAB$SL_FNA];
: 619      0680      4      CCB [LUB$B_RSL] = .FAB [FAB$SB_FNS];
: 620      0681      2      !<BLF/PAGE>

```

```

: 622      0682      2
: 623      0683      2
: 624      0684      2
: 625      0685      2
: 626      0686      2
: 627      0687      2
: 628      0688      2
: 629      0689      2
: 630      0690      2
: 631      0691      2
: 632      0692      2
: 633      0693      2
: 634      0694      2
: 635      0695      2
: 636      0696      2
: 637      0697      2
: 638      0698      2
: 639      0699      2

!+
! Do a $PARSE on the file to see if the file is a network file.  If
! so, we will set FAB$V_SQ0 and not enable RFA cacheing.  Otherwise,
! we'll leave SQ0 clear so that RFA cacheing can be allowed.
!-

FAB [FAB$L_NAM] = 0;
IF $PARSE (FAB = FAB [0,0,0,0])
THEN
    BEGIN
    BIND
        FAB_DEV = FAB [FAB$L_DEV]: BLOCK [4, BYTE];
    IF .FAB_DEV [DEV$V_NET]
    THEN
        FAB [FAB$V_SQ0] = 1;
    END;
FAB [FAB$L_STS] = 0;      ! Hide error, if any
FAB [FAB$L_NAM] = NAM [0,0,0,0];

```

```
641 0700 2
642 0701 2
643 0702 2
644 0703 2
645 0704 2
646 0705 2
647 0706 2
648 0707 2
649 0708 2
650 0709 2
651 0710 2
652 0711 2
653 0712 2
654 0713 2
655 0714 2
656 0715 2
657 0716 2
658 0717 2
659 0718 2
660 0719 2
661 0720 2
662 0721 2
663 0722 2
664 0723 2
665 0724 2
666 0725 2
667 0726 2
668 0727 2
669 0728 2
670 0729 2
671 0730 2
672 0731 2
673 0732 2
674 0733 2
675 0734 2
676 0735 2
677 0736 2
678 0737 2
679 0738 2
680 0739 2
681 0740 2
682 0741 2
683 0742 2
684 0743 2
685 0744 2
686 0745 2
687 0746 2
688 0747 2
689 0748 2
690 0749 2
691 0750 2
692 0751 2
693 0752 2
694 0753 2
695 0754 2
696 0755 2
697 0756 2

+
READONLY
Set functions which may be done subsequently (FAB$B_FAC).
If not READONLY, permit GET, PUT, TRUNCATE (via TPTT), UPDATE and DELETE.
If READONLY, set LUB$V_READ_ONLY bit and use RMS default functions
which can be done subsequently, namely just GETs.
-

IF .OPEN_ADR [OPEN$K_READONLY]
THEN
  BEGIN
    CCB [LUB$V_READ_ONLY] = 1;
  END
ELSE
  IF (.FAB [FAB$B_FAC] EQLU 0)
  THEN
    FAB [FAB$B_FAC] = FAB$M_GET + FAB$M_PUT + FAB$M_TRN + FAB$M_DEL + FAB$M_UPD;

+
ACCESS
-

+
If LUB$L_LOG_RECNO is zero, then this is not a default open of
a direct access file, so set the record number to 1. Otherwise,
leave it alone because it has already been set by FOR$IO_BEG.
-

IF .CCB [LUB$L_LOG_RECNO] EQL 0
THEN
  CCB [LUB$L_LOG_RECNO] = 1;

FAB [FAB$V_NEF] = 1;          ! inhibit EOF positioning on MT

CASE .OPEN_ADR [OPEN$K_ACCESS] FROM 0 TO OPEN$K_ACC_KEY OF
  SET
    [OPEN$K_ACC_DIR] :          ! ACCESS = 'DIRECT'
      BEGIN
        CCB [LUB$V_DIRECT] = 1;
        FAB [FAB$V_SQO] = 0;    ! May have been set earlier
        CCB [RAB$B_RAC] = RAB$C_KEY;
        CCB [RAB$L_KBF] = CCB [LUB$L_LOG_RECNO];
        CCB [RAB$B_KSZ] = 0;
        CCB [RAB$V_UIF] = 1;    ! Update on $PUT
      END;
    [0, OPEN$K_ACC_SEQ] :      ! omitted or ACCESS = 'SEQUENTIAL'
      BEGIN
        CCB [LUB$V_SEQUENTIAL] = 1;
        CCB [RAB$B_RAC] = RAB$C_SEQ;
      END;
    [OPEN$K_ACC_APP] :        ! ACCESS = 'APPEND'
      BEGIN
        IF .CCB [LUB$V_READ_ONLY]
        THEN
```

```

: 698      0757      3      $FOR$$$SIGNAL_STO (FOR$K_INCOPECLO);
: 699      0758      3      CCB [RAB$V_EOF] = 1;
: 700      0759      3      CCB [LUB$V_APPEND] = 1;
: 701      0760      3      FAB [FAB$V_NEF] = 0;          ! don't inhibit EOF positioning on MT
: 702      0761      3      CCB [RAB$B_RAC] = RAB$C_SEQ;
: 703      0762      3      END;
: 704      0763      3
: 705      0764      3      [OPEN$K_ACC_KEY] :          ! ACCESS = 'KEYED'
: 706      0765      3      BEGIN
: 707      0766      3      FAB [FAB$V_SQO] = 0;          ! May have been set earlier
: 708      0767      3      CCB [RAB$B_RAC] = RAB$C_KEY;
: 709      0768      3      CCB [RAB$B_KRF] = 0;
: 710      0769      3      CCB [LUB$V_KEYED] = 1;        ! So we know later
: 711      0770      3      END;
: 712      0771      3
: 713      0772      3      [OUTRANGE] :
: 714      0773      3      $FOR$$$SIGNAL_STO (FOR$K_INVARGFOR);
: 715      0774      3      TES;
: 716      0775      3
: 717      0776      2      !<BLF/PAGE>
```

```
719 0777 2 !
720 0778 2
721 0779 2
722 0780 2 TYPE
723 0781 2
724 0782 2
725 0783 2 CASE .OPEN_ADR [OPENS$K_TYPE] FROM 0 TO OPENS$K_TYP_UNK OF
726 0784 2 SET
727 0785 2
728 0786 2 [OPENS$K_TYP_OLD] : ! TYPE = 'OLD'
729 0787 2 CCB [LUB$V_OLD_FILE] = 1;
730 0788 2
731 0789 2 [0, OPENS$K_TYP_NEW] : ! omitted or TYPE = 'NEW'
732 0790 2 BEGIN
733 0791 2 IF NOT .CCB [LUB$V_OLD_FILE] ! Could have been set by FDBSET
734 0792 2 THEN
735 0793 2 IF .CCB [LUB$V_READ_ONLY] OR
736 0794 2 .CCB [LUB$V_APPEND]
737 0795 2 THEN
738 0796 2 $FOR$$$SIGNAL_STO (FOR$K_INCOPECLO);
739 0797 2 END;
740 0798 2
741 0799 2 [OPENS$K_TYP_SCR] : ! TYPE = 'SCRATCH'
742 0800 2 BEGIN
743 0801 2 CCB [LUB$V_SCRATCH] = 1;
744 0802 2 FAB [FAB$V_TMD] = 1;
745 0803 2 IF .CCB [LUB$V_READ_ONLY] OR
746 0804 2 .CCB [LUB$V_APPEND]
747 0805 2 THEN
748 0806 2 $FOR$$$SIGNAL_STO (FOR$K_INCOPECLO);
749 0807 2 END;
750 0808 2
751 0809 2 [OPENS$K_TYP_UNK] : ! TYPE = 'UNKNOWN'
752 0810 2 BEGIN
753 0811 2 FAB [FAB$V_CIF] = 1;
754 0812 2 IF .CCB [LUB$V_READ_ONLY]
755 0813 2 THEN
756 0814 2 $FOR$$$SIGNAL_STO (FOR$K_INCOPECLO);
757 0815 2 END;
758 0816 2
759 0817 2 [OUTRANGE] :
760 0818 2 $FOR$$$SIGNAL_STO (FOR$K_INVARGFOR);
761 0819 2 TES;
762 0820 2
763 0821 2 !<BLF/PAGE>
```

```
765 0822 2 !
766 0823
767 0824
768 0825
769 0826
770 0827
771 0828
772 0829
773 0830
774 0831
775 0832
776 0833
777 0834
778 0835
779 0836
780 0837
781 0838
782 0839
783 0840
784 0841
785 0842
786 0843
787 0844
788 0845
789 0846
790 0847
791 0848
792 0849
793 0850
794 0851
795 0852
796 0853
797 0854
798 0855
799 0856
800 0857
801 0858
802 0859
803 0860
804 0861
805 0862
806 0863
807 0864
808 0865
809 0866
810 0867
811 0868
812 0869
813 0870
814 0871
815 0872
816 0873
817 0874
818 0875

!
DISPOSE
Set bits in LUB to indicate DISPOSE parameters. Do not allow
deletion of READONLY or SCRATCH files, printing or submitting of
SCRATCH files, or saving of SCRATCH files.

SELECT .OPEN_ADR [OPEN$K_DISPOSE] OF
SET
[0] :
; ! omitted, do nothing
[OPEN$K_DIS_SAV] : ! DISPOSE = 'SAVE'
IF .CCB [LUB$V_SCRATCH] THEN $FOR$$SIGNAL_STO (FOR$K_INCOPECLO);
[OPEN$K_DIS_DEL, OPEN$K_DIS_PRDE, OPEN$K_DIS_SUDE] :
! DISPOSE = 'DELETE', 'PRINT/DELETE', 'SUBMIT/DELETE'
BEGIN
IF .CCB [LUB$V_READ_ONLY]
THEN
$FOR$$SIGNAL_STO (FOR$K_INCOPECLO);
CCB [LUB$V_DELETE] = 1;
END;
[OPEN$K_DIS_PRI, OPEN$K_DIS_PRDE] :
! DISPOSE = 'PRINT', 'PRINT/DELETE'
BEGIN
IF .CCB [LUB$V_SCRATCH] THEN $FOR$$SIGNAL_STO (FOR$K_INCOPECLO);
CCB [LUB$V_PRINT] = 1;
END;
[OPEN$K_DIS_SUB, OPEN$K_DIS_SUDE] :
! DISPOSE = 'SUBMIT', 'SUBMIT/DELETE'
BEGIN
IF .CCB [LUB$V_SCRATCH]
THEN
$FOR$$SIGNAL_STO (FOR$K_INCOPECLO)
ELSE
CCB [LUB$V_SUBMIT] = 1;
END;
[OTHERWISE] :
$FOR$$SIGNAL_STO (FOR$K_INVARGFOR);
YES;
!<BLF/PAGE>
```

```

0820 0876
0821 0877
0822 0878
0823 0879
0824 0880
0825 0881
0826 0882
0827 0883
0828 0884
0829 0885
0830 0886
0831 0887
0832 0888
0833 0889
0834 0890
0835 0891
0836 0892
0837 0893
0838 0894
0839 0895
0840 0896
0841 0897
0842 0898
0843 0899
0844 0900
0845 0901
0846 0902
0847 0903
0848 0904
0849 0905
0850 0906

```

```

!
FORM
CASE .OPEN_ADR [OPENS$K_FORM] FROM OPENS$K_FOR_UN$ TO OPENS$K_FOR_UNF OF
  SET
    [OPENS$K_FOR_UN$] :
      ; ! unspecified, used by default OPEN only
    [0] : ! omitted
      IF .CCB [LUB$V_DIRECT] OR .CCB [LUB$V_KEYED]
      THEN
        CCB [LUB$V_UNFORMAT] = 1
      ELSE
        CCB [LUB$V_FORMATTED] = 1;
    [OPENS$K_FOR_FOR] : ! FORM = 'FORMATTED'
      CCB [LUB$V_FORMATTED] = 1;
    [OPENS$K_FOR_UNF] : ! FORM = 'UNFORMATTED'
      CCB [LUB$V_UNFORMAT] = 1;
    [OUTRANGE] :
      $FOR$$SIGNAL_STO (FOR$K_INVARGFOR);
  TES;
!<BLF/PAGE>

```

```
852      0907      !  
853      0908      !  
854      0909      !  
855      0910      ! RECORDTYPE  
856      0911      !  
857      0912      !  
858      0913      !  
859      0914      ! CASE .OPEN_ADR [OPEN$K_RECORDTY] FROM 0 TO OPEN$K_REC_STMLF OF  
860      0915      ! SET  
861      0916      ! [0] : ! omitted  
862      0917      !  
863      0918      !  
864      0919      ! Do nothing right now. We have insufficient information  
865      0920      ! to determine the recordtype. Wait until the organization  
866      0921      ! has been determined.  
867      0922      !  
868      0923      !  
869      0924      !  
870      0925      !  
871      0926      ! [OPEN$K_REC_FIX] : ! RECORDTYPE = 'FIXED'  
872      0927      ! BEGIN  
873      0928      ! CCB [LUB$V_FIXED] = 1;  
874      0929      ! FAB [FAB$B_RFM] = FAB$C_FIX;  
875      0930      ! END;  
876      0931      !  
877      0932      ! [OPEN$K_REC_VAR] : ! RECORDTYPE = 'VARIABLE'  
878      0933      ! BEGIN  
879      0934      ! FAB [FAB$B_RFM] = FAB$C_VAR;  
880      0935      ! END;  
881      0936      !  
882      0937      ! [OPEN$K_REC_SEGM] : ! RECORDTYPE = 'SEGMENTED'  
883      0938      ! BEGIN  
884      0939      !  
885      0940      ! IF .CCB [LUB$V_DIRECT] OR .CCB [LUB$V_KEYED] OR .CCB [LUB$V_FORMATTED]  
886      0941      ! THEN  
887      0942      ! $FOR$$SIGNAL_STO (FOR$K_INCOPECLO);  
888      0943      !  
889      0944      ! FAB [FAB$B_RFM] = FAB$C_VAR;  
890      0945      ! CCB [LUB$V_SEGMENTED] = 1;  
891      0946      ! END;  
892      0947      !  
893      0948      ! [OPEN$K_REC_STM] : ! RECORDTYPE = 'STREAM'  
894      0949      ! BEGIN  
895      0950      ! FAB [FAB$B_RFM] = FAB$C_STM;  
896      0951      ! END;  
897      0952      !  
898      0953      ! [OPEN$K_REC_STMCR] : ! RECORDTYPE = 'STREAM_CR'  
899      0954      ! BEGIN  
900      0955      ! FAB [FAB$B_RFM] = FAB$C_STMCR;  
901      0956      ! END;  
902      0957      !  
903      0958      ! [OPEN$K_REC_STMLF] : ! RECORDTYPE = 'STREAM_LF'  
904      0959      ! BEGIN  
905      0960      ! FAB [FAB$B_RFM] = FAB$C_STMLF;  
906      0961      ! END;  
907      0962      !  
908      0963      ! [OUTRANGE] :
```

FOR\$OPEN_DEFLT FORTRAN default open
1-098

G 10
16-Sep-1984 00:37:00
14-Sep-1984 12:32:16

VAX-11 Bliss-32 V4.0-742
[FORRTL.SRC]FOROPENDE.B32;1

Page 21
(11)

```

: 909      0964 2      $FOR$SIGNAL_STO (FOR$K_INVARGFOR);
: 910      0965 2      TES;
: 911      0966 2
: 912      0967 2 !<BLF/PAGE>
```

```
..... 914      0968      :  
..... 915      0969      :  
..... 916      0970      :  
..... 917      0971      :  
..... 918      0972      :  
..... 919      0973      :  
..... 920      0974      :  
..... 921      0975      :  
..... 922      0976      :  
..... 923      0977      :  
..... 924      0978      :  
..... 925      0979      :  
..... 926      0980      :  
..... 927      0981      :  
..... 928      0982      :  
..... 929      0983      :  
..... 930      0984      :  
..... 931      0985      :  
..... 932      0986      :  
..... 933      0987      :  
..... 934      0988      :  
..... 935      0989      :  
..... 936      0990      :  
..... 937      0991      :  
..... 938      0992      :  
..... 939      0993      :  
..... 940      0994      :  
..... 941      0995      :  
..... 942      0996      :  
..... 943      0997      :  
..... 944      0998      :  
..... 945      0999      :  
..... 946      1000      :  
..... 947      1001      :  
      :  
      !+  
      !- CARRIAGECONTROL  
      :  
      CASE .OPEN_ADR [OPENS$CARRIAGE] FROM 0 TO OPENS$CAR_NON OF  
      SET  
      [0] : ! omitted  
      IF .CCB [LUB$V_FORMATTED] THEN FAB [FAB$V_FTN] = 1;  
      [OPENS$CAR_FOR] : ! CARRIAGECONTROL = 'FORTRAN'  
      FAB [FAB$V_FTN] = 1;  
      [OPENS$CAR_LIS] : ! CARRIAGECONTROL = 'LIST'  
      FAB [FAB$V_CR] = 1;  
      [OPENS$CAR_NON] :  
      : ! CARRIAGECONTROL = 'NONE', do nothing  
      [OUTRANGE] :  
      $FOR$$$IGNAL_STO (FOR$K_INVARGFOR);  
      TES;  
      :+  
      : Store FAB$B_RAT so we can "restore" it if we find we've  
      : opened a process-permanent file.  
      :-  
      ORIG_RAT = .FAB [FAB$B_RAT];  
      !<BLF/PAGE>
```

```

949      1002      !
950      1003      !
951      1004      !
952      1005      !
953      1006      !
954      1007      !
955      1008      !
956      1009      !
957      1010      !
958      1011      !
959      1012      !
960      1013      !
961      1014      !
962      1015      !
963      1016      !
964      1017      !
965      1018      !
966      1019      !
967      1020      !
968      1021      !
969      1022      !
970      1023      !
971      1024      !
972      1025      !
973      1026      !
974      1027      !
975      1028      !
976      1029      !
977      1030      !
978      1031      !
979      1032      !
980      1033      !
981      1034      !
982      1035      !
983      1036      !
984      1037      !
985      1038      !
986      1039      !
987      1040      !
988      1041      !
989      1042      !
990      1043      !
991      1044      !
992      1045      !
993      1046      !
994      1047      !
995      1048      !
996      1049      !
997      1050      !
998      1051      !
999      1052      !
1000     1053      !
1001     1054      !
1002     1055      !
1003     1056      !
1004     1057      !
1005     1058      !

      !+
      !- ORGANIZATION
      CCB [LUB$V_NOTSEQORG] = 1;                ! Assume not sequential organization
      CASE .OPEN_ADR [OPEN$K_ORGANIZA] FROM 0 TO OPEN$K_ORG_IDX OF
      SET
      [0, OPEN$K_ORG_SEQ] :                      ! omitted or ORGANIZATION = ;SEQUENTIAL'
      BEGIN
      IF .CCB [LUB$V_DIRECT] AND .FAB [FAB$B_RFM] EQLU FAB$C_VAR THEN $FOR$$SIGNAL_STO (FOR$K_INCOPECL
      ;
      IF .CCB [LUB$V_KEYED] AND .OPEN_ADR [OPEN$K_ORGANIZA] NEQ 0
      THEN
      $FOR$$SIGNAL_STO (FOR$K_INCOPECLO);

      FAB [FAB$B_ORG] = FAB$C_SEQ;
      CCB [LUB$V_NOTSEQORG] = 0;                ! So ENDFILE will know its sequential
      END;

      [OPEN$K_ORG_REL] :                          ! ORGANIZATION = 'RELATIVE'
      BEGIN
      IF .CCB [LUB$V_SEGMENTED] OR .CCB [LUB$V_KEYED] THEN $FOR$$SIGNAL_STO (FOR$K_INCOPECLO);

      FAB [FAB$B_ORG] = FAB$C_REL;
      END;

      [OPEN$K_ORG_IDX] :                          ! ORGANIZATION = 'INDEXED'
      BEGIN
      IF .CCB [LUB$V_DIRECT] OR .CCB [LUB$V_APPEND] OR .CCB [LUB$V_SEGMENTED]
      THEN
      $FOR$$SIGNAL_STO (FOR$K_INCOPECLO);

      FAB [FAB$B_ORG] = FAB$C_IDX;
      END;

      [OUTRANGE] :
      $FOR$$SIGNAL_STO (FOR$K_INVARGFOR);
      TES;

      !+
      !- Verify that user didn't ask for a non-sequential stream file.
      !-
      IF .CCB [LUB$V_NOTSEQORG] AND
      ONE_OF (.FAB [FAB$B_RFM], FAB$C_STM, FAB$C_STMCR, FAB$C_STMLF)
      THEN
      $FOR$$SIGNAL_STO (FOR$K_INCOPECLO);
```

```
1006 1059 2
1007 1060 2
1008 1061 2
1009 1062 2
1010 1063 2
1011 1064 2
1012 1065 2
1013 1066 2
1014 1067 2
1015 1068 2
1016 1069 2
1017 1070 2
1018 1071 2
1019 1072 2
1020 1073 2
1021 1074 2
1022 1075 2
1023 1076 2
1024 1077 2
1025 1078 2
1026 1079 2
1027 1080 2
1028 1081 2
1029 1082 2
1030 1083 2
1031 1084 2
1032 1085 2
1033 1086 2
1034 1087 2
1035 1088 2
1036 1089 2
1037 1090 2
1038 1091 2
1039 1092 2
1040 1093 2
1041 1094 2
1042 1095 2
1043 1096 2
1044 1097 2
1045 1098 2
1046 1099 2
1047 1100 2 !<BLF/PAGE>

+ RECORDTYPE continued
We now have enough information to determine the initial recordtype
if it was omitted.
-

IF .OPEN_ADR [OPEN$K_RECORDTY] EQL 0
THEN
    IF .FAB [FAB$B_ORG] EQL FAB$C_REL OR .FAB [FAB$B_ORG] EQL FAB$C_IDX OR .CCB [LUB$V_DIRECT] OR .CCB [
        LUB$V_REYED]
    THEN
        BEGIN
            FAB [FAB$B_RFM] = FAB$C_FIX;
            CCB [LUB$V_FIXED] = 1;
        END
    ELSE
        BEGIN
            FAB [FAB$B_RFM] = FAB$C_VAR;

            IF .CCB [LUB$V_UNFORMAT] THEN CCB [LUB$V_SEGMENTED] = 1;

        END;

+ SHARED
If SHARED, indicate user provided record interlock (UPI) (for SEQUENTIAL ORG only)
If not SHARED, RMS defaults is read, sharing only if READONLY, else no sharing.
-

IF .OPEN_ADR [OPEN$K_SHARED]
THEN
    BEGIN
        FAB [FAB$B_SHR] = FAB$M_SHRGET + FAB$M_SHRPUT + FAB$M_SHRUPD + FAB$M_SHRDEL;

        IF NOT .CCB [LUB$V_NOTSEQORG] ! Sequential only, set UPI
        THEN
            FAB [FAB$V_UPI] = 1;

    END;

!<BLF/PAGE>
```

```
1049      1101      !
1050      1102      !
1051      1103      !+ KEY
1052      1104      !-
1053      1105      !
1054      1106      !
1055      1107      IF .OPEN_ADR [OPENS$KEY] NEQ 0
1056      1108      THEN
1057      1109      BEGIN
1058      1110      LOCAL
1059      1111      KEY_DEFN : REF BLOCK [12, BYTE],      ! Key definition
1060      1112      KEY_NUM,      ! Number of current key
1061      1113      KEY_COUNT,      ! Total number of keys defined
1062      1114      XAB_ADDR;      ! Address of newly allocated KEY XAB
1063      1115
1064      1116      IF .FAB [FABS$ORG] NEQ FAB$C_IDX THEN $FOR$$SIGNAL_STO (FOR$K_INCOPECLO);
1065      1117
1066      1118      KEY_DEFN = .OPEN_ADR [OPENS$KEY];
1067      1119      KEY_COUNT = .KEY_DEFN [OPENS$INFO];
1068      1120      KEY_DEFN = .KEY_DEFN + XUPVAL;
1069      1121
1070      1122      IF .KEY_COUNT MOD 3 NEQ 0 THEN $FOR$$SIGNAL_STO (FOR$K_INVARGFOR);
1071      1123
1072      1124      KEY_COUNT = .KEY_COUNT/3;
1073      1125
1074      1126      !+
1075      1127      !- Loop through key definitions, and set up the key XABs.
1076      1128
1077      1129
1078      1130      INCR KEY_NUM FROM 0 TO .KEY_COUNT - 1 DO
1079      1131      BEGIN
1080      1132      XAB_ADDR = FOR$$GET_VM (OPENS$XAB_SIZE);
1081      1133      KEY_XAB [XABS$NXT] = .XAB_ADDR;
1082      1134      KEY_XAB = .XAB_ADDR;
1083      1135
1084      1136      !+
1085      1137      !- Fill in KEY XAB fields
1086      1138
1087      1139
1088      1140      CHSFILL (0, OPENS$XAB_SIZE, .KEY_XAB);
1089      1141      KEY_XAB [XABS$COD] = XABS$KEY;
1090      1142      KEY_XAB [XABS$BLN] = XABS$KEYLEN;
1091      1143
1092      1144      !+
1093      1145      !- Calculate key position and width
1094      1146
1095      1147
1096      1148
1097      1149      IF .KEY_DEFN [OPENS$KEY_LO] LEQ 0 OR
1098      1150      .KEY_DEFN [OPENS$KEY_LO] GTR 32767 OR
1099      1151      .KEY_DEFN [OPENS$KEY_HI] GTR 32767 OR
1100      1152      .KEY_DEFN [OPENS$KEY_HI] LSS .KEY_DEFN [OPENS$KEY_LO]
1101      1153      THEN
1102      1154      $FOR$$SIGNAL_STO (FOR$K_INVKEYSPE);
1103      1155
1104      1156      KEY_XAB [XABS$POS0] = .KEY_DEFN [OPENS$KEY_LO] - 1;
1105      1157      KEY_XAB [XABS$SIZ0] =
```

```
1106      1158      5      BEGIN
1107      1159      3
1108      1160      3      LOCAL
1109      1161      3      SIZE;
1110      1162      3
1111      1163      3      SIZE = .KEY_DEFN [OPENS$KEY_HI] - .KEY_DEFN [OPENS$KEY_LO] + 1;
1112      1164      3
1113      1165      3      IF .SIZE GTR 255 THEN $FOR$$SIGNAL_STO (FOR$K_INVKEYSPE);
1114      1166      3
1115      1167      3      .SIZE
1116      1168      4      END;
1117      1169      4      KEY_XAB [OPENS$W_POS0] = .KEY_XAB [XAB$W_POS0];
1118      1170      4      KEY_XAB [OPENS$B_SIZE0] = .KEY_XAB [XAB$B_SIZE0];
1119      1171      3      KEY_XAB [XAB$B_DTP] = (SELECTONE .KEY_DEFN [OPENS$B_DTYPE] OF
1120      1172      3      SET
1121      1173      3      [0, DSC$K_DTYPE_T] : XAB$C_STG;
1122      1174      3      [DSC$K_DTYPE_WU] : XAB$C_BN2;
1123      1175      3      [DSC$K_DTYPE_W] : XAB$C_IN2;
1124      1176      3      [DSC$K_DTYPE_LU] : IF .KEY_XAB [XAB$B_SIZE0] EQL 4 THEN XAB$C_BN4 ELSE XAB$C_BN2;
1125      1177      3      [DSC$K_DTYPE_L] : IF .KEY_XAB [XAB$B_SIZE0] EQL 4 THEN XAB$C_IN4 ELSE XAB$C_IN2;
1126      1178      3      [OTHERWISE] :
1127      1179      6      BEGIN
1128      1180      6      $FOR$$SIGNAL_STO (FOR$K_INVARGFOR);
1129      1181      3      END;
1130      1182      4      TES);
1131      1183      4      KEY_XAB [OPENS$B_KTYPE] = .KEY_XAB [XAB$B_DTP];
1132      1184      4
1133      1185      4      IF .KEY_NUM NEQ 0
1134      1186      4      THEN
1135      1187      3      BEGIN
1136      1188      3      KEY_XAB [XAB$V_CHG] = 1;
1137      1189      3      KEY_XAB [XAB$V_DUP] = 1;
1138      1190      4      END;
1139      1191      4
1140      1192      4      KEY_XAB [XAB$B_REF] = .KEY_NUM;
1141      1193      4      KEY_DEFN = .KEY_DEFN + (3*ZUPVAL); ! Go to next definition
1142      1194      3      END;
1143      1195      3
1144      1196      2      END;
1145      1197      2
1146      1198      2
1147      1199      2      !+
1148      1200      2      BLANK
1149      1201      2      If user specifies BLANK='NULL' then set LUB$V_NULLBLNK
1150      1202      2      else leave it alone.
1151      1203      2      -
1152      1204      2      CASE .OPEN_ADR [OPENS$K_BLANK] FROM 0 TO OPENS$K_BLK_NUL OF
1153      1205      2      SET
1154      1206      2
1155      1207      2      [0, OPENS$K_BLK_ZER] :
1156      1208      2      ;
1157      1209      2      ! Do nothing, ZERO is the default
1158      1210      2
1159      1211      2      [OPENS$K_BLK_NUL] :
1160      1212      2      [(CB [LUB$V_NULLBLNK] = 1;
1161      1213      2
1162      1214      2      [OUTRANGE] :
1162      1214      2      $FOR$$SIGNAL_STO (FOR$K_INVARGFOR);
```

FORSSOPEN_DEFLT FORTRAN default open
1-098

M 10
16-Sep-1984 00:37:00
14-Sep-1984 12:32:16

VAX-11 Bliss-32 V4.0-742
[FORRTL.SRC]FOROPENDE.B32;1

Page 27
(14)

: 1163 1215 2 TES:
: 1164 1216 2
: 1165 1217 2 !<BLF/PAGE>

```
1167 1218 2 !
1168 1219 2
1169 1220 2
1170 1221 2
1171 1222 2
1172 1223 2
1173 1224 2
1174 1225 2
1175 1226 2
1176 1227 2
1177 1228 2
1178 1229 2
1179 1230 2
1180 1231 2
1181 1232 2
1182 1233 2
1183 1234 2
1184 1235 2
1185 1236 2
1186 1237 2
1187 1238 2
1188 1239 2
1189 1240 2
1190 1241 2
1191 1242 2
1192 1243 2
1193 1244 2
1194 1245 2
1195 1246 2
1196 1247 2
1197 1248 2
1198 1249 2
1199 1250 2
1200 1251 2
1201 1252 2
1202 1253 2
1203 1254 2
1204 1255 2
1205 1256 2
1206 1257 2
1207 1258 2
1208 1259 2
1209 1260 2
1210 1261 2
1211 1262 2
1212 1263 2
1213 1264 2
1214 1265 2
1215 1266 2
1216 1267 2
1217 1268 2
1218 1269 2
1219 1270 2
1220 1271 2
1221 1272 2
1222 1273 2
1223 1274 2

+ RECORDSIZE
Set maximum record size (FAB$W_MRS) if fixed, relative, or indexed.
Set V_DEFAULT_SIZE if omitted. Set LUB$W_RBUF_SIZE to record size.
Default is 128 for unformatted fixed length, 2044 for unformatted
variable length (4 bytes for RMS control info to make total 2048),
and 133 for formatted (line printer width) or unspecified (ENDFILE
default OPEN).

V_DEFAULT_SIZE = 0; ! assume user specifies

SELECTONEU .OPEN_ADR [OPEN$K_RECORDS] OF
SET
[0] :
+ If this is a fixed length or relative file, and
is not known to exist, RECORDSIZE must be given, else
error FOR$_INCRECLEN.
-
IF .CCB [LUB$W_RBUF_SIZE] EQLU 0
THEN
BEGIN
IF NOT .CCB [LUB$V_OLD_FILE] AND (.CCB [LUB$V_FIXED]
OR .FAB [FAB$B_ORG] EQL FAB$C_REL)
THEN
$FOR$$SIGNAL_STO (FOR$K_INCRECLEN);

CCB [LUB$W_RBUF_SIZE] = (
IF .CCB [LUB$V_UNFORMAT] ! unformatted
THEN
IF .CCB [LUB$V_FIXED]
THEN 128 ! fixed
ELSE 2044 ! variable
ELSE 133 ! formatted or unspecified (ENDFILE default open)
);
V_DEFAULT_SIZE = 1; ! user took the default
END;

[1 TO 32767] :
BEGIN
LOCAL
T;

T = .OPEN_ADR [OPEN$K_RECORDS] + (IF .CCB [LUB$V_UNFORMAT] THEN %UPVAL ELSE 1) !
+ (IF .CCB [LUB$V_SEGMENTED] THEN 2 ELSE 0);

IF .T GTRU 32767 THEN $FOR$$SIGNAL_STO (FOR$K_INCRECLEN);
```

```

1224      1275      3
1225      1276      3
1226      1277      3
1227      1278      3
1228      1279      3
1229      1280      3
1230      1281      3
1231      1282      3
1232      1283      3
1233      1284      3
1234      1285      3
1235      1286      2
1236      1287      2
1237      1288      2 !<BLF/PAGE>

      CCB [LUB$W_RBUF_SIZE] = .T;
      END;

      [OTHERWISE] :
      $FOR$$SIGNAL_STO (FOR$K_INCRECLEN);
      TES;

      IF .CCB [LUB$V_FIXED]
      OR (.FAB [FAB$B_ORG] EQLU FAB$C_REL)
      OR (.FAB [FAB$B_ORG] EQLU FAB$C_IDX)
      THEN FAB [FAB$W_MRS] = .CCB [LUB$W_RBUF_SIZE];

```

```
1239 1289 2 !
1240 1290
1241 1291
1242 1292
1243 1293
1244 1294
1245 1295
1246 1296
1247 1297
1248 1298
1249 1299
1250 1300
1251 1301
1252 1302
1253 1303
1254 1304
1255 1305
1256 1306
1257 1307
1258 1308
1259 1309
1260 1310
1261 1311
1262 1312
1263 1313
1264 1314
1265 1315
1266 1316
1267 1317
1268 1318
1269 1319
1270 1320
1271 1321
1272 1322
1273 1323
1274 1324
1275 1325
1276 1326
1277 1327
1278 1328
1279 1329
1280 1330
1281 1331
1282 1332 2 !<BLF/PAGE>

+ INITIALSIZE
Only set if specified in explicit OPEN, since may be set by FDBSET on default OPEN.

IF .OPEN_ADR [OPEN$K_INITIALS] NEQ 0
THEN
  BEGIN
    FAB [FAB$L_ALQ] = ABS (.OPEN_ADR [OPEN$K_INITIALS]);
    FAB [FAB$V_CBT] = 1;
  END;

+ EXTENDSIZE
Only set if specified explicitly in explicit OPEN, since FDBSET could set on default open.

IF .OPEN_ADR [OPEN$K_EXTENDSI] NEQU 0
THEN
  IF ABS (.OPEN_ADR [OPEN$K_EXTENDSI]) LSSU 1^16
  THEN
    FAB [FAB$V_DEQ] = ABS (.OPEN_ADR [OPEN$K_EXTENDSI])
  ELSE
    $FOR$$$SIGNAL_STO (FOR$K_KEYVALERR);

+ NOSPANBLOCKS

FAB [FAB$V_BLK] = .OPEN_ADR [OPEN$K_NOSPANBL];

+ MAXREC
Only set if explicitly passed by OPEN statement, since
DEFINE FILE could have pre-set it if this is default open.

IF .OPEN_ADR [OPEN$K_MAXREC] NEQU 0 THEN CCB [LUB$L_REC_MAX] = .OPEN_ADR [OPEN$K_MAXREC];

FAB [FAB$L_MRN] = .CCB [LUB$L_REC_MAX];
```

```
1284 1333 2 !
1285 1334
1286 1335
1287 1336
1288 1337
1289 1338
1290 1339
1291 1340
1292 1341
1293 1342
1294 1343
1295 1344
1296 1345
1297 1346
1298 1347
1299 1348
1300 1349
1301 1350
1302 1351
1303 1352
1304 1353
1305 1354
1306 1355
1307 1356
1308 1357
1309 1358
1310 1359
1311 1360
1312 1361 2 !<BLF/PAGE>

+ BLOCKSIZE
Set BLOCKSIZE (used for magtape only), multi-block count (sequential org only)
and bucket size (relative/indexed only).

SELECTONEU .OPEN_ADR [OPEN$K_BLOCKSIZ] OF
SET
[0] :
; ! Use process/system defaults

[1 TO 65535] :
BEGIN
FAB [FAB$W_BLS] = .OPEN_ADR [OPEN$K_BLOCKSIZ];
CCB [RAB$B_MBC] = (.OPEN_ADR [OPEN$K_BLOCKSIZ] + 511)/512;
FAB [FAB$B_BKS] = .CCB [RAB$B_MBC];
IF .FAB [FAB$B_BKS] GTRU 63 !-RMS limit
THEN
FAB [FAB$B_BKS] = 63;
END;

[OTHERWISE] :
$FOR$$SIGNAL_STO (FOR$K_KEYVALERR);
TES;
```

```
1314      1362      !
1315      1363      !
1316      1364      !
1317      1365      !
1318      1366      !
1319      1367      !
1320      1368      !
1321      1369      !
1322      1370      !
1323      1371      !
1324      1372      !
1325      1373      !
1326      1374      !
1327      1375      !
1328      1376      !
1329      1377      !
1330      1378      !
1331      1379      !
1332      1380      !
1333      1381      !
1334      1382      !
1335      1383      !
1336      1384      !
1337      1385      !
1338      1386      !
1339      1387      !
1340      1388      !
1341      1389      !
1342      1390      !
1343      1391      !
1344      1392      !
1345      1393      !
1346      1394      !
1347      1395      !

      !
      !+
      ! BUFFERCOUNT
      ! Only set if explicitly passed by OPEN statement since FDBSET could
      ! have pre-set it if this is a default open.
      !-

      SELECTONEU .OPEN_ADR [OPEN$K_BUFFERCO] OF
      SET
      [0] :
      ;
      [1 TO 127] :
      CCB [RAB$B_MBF] = .OPEN_ADR [OPEN$K_BUFFERCO];
      [OTHERWISE] :
      $FOR$$SIGNAL_STO (FOR$K_KEYVALERR);
      TES;

      !+
      ! ASSOCIATEVARIABLE
      !-

      IF .OPEN_ADR [OPEN$K_ASSOCIAT] NEQA 0
      THEN
      BEGIN
      CCB [LUB$A_ASSOC_VAR] = .OPEN_ADR [OPEN$K_ASSOCIAT];
      IF .OPEN_ADR [OPEN$K_ASSOC_L] THEN CCB [LUB$V_ASS_VAR_L] = 1
      END;
```

```
1349      1396      2 !
1350      1397      2
1351      1398      2
1352      1399      2
1353      1400      2
1354      1401      2
1355      1402      2
1356      1403      2
1357      1404      2
1358      1405      2
1359      1406      2
1360      1407      2
1361      1408      2
1362      1409      2
1363      1410      2
1364      1411      2
1365      1412      2
1366      1413      2
1367      1414      2
1368      1415      2
1369      1416      2
1370      1417      2
1371      1418      2
1372      1419      2
1373      1420      2
1374      1421      2
1375      1422      2
1376      1423      2
1377      1424      2
1378      1425      2
1379      1426      2
1380      1427      2
1381      1428      2
1382      1429      2
1383      1430      2
1384      1431      2
1385      1432      2
1386      1433      2
1387      1434      2
1388      1435      2
1389      1436      2
1390      1437      2
1391      1438      2
1392      1439      2
1393      1440      2
1394      1441      2
1395      1442      2
1396      1443      2
1397      1444      2
1398      1445      2
1399      1446      2

      !+
      ! USEROPEN
      ! If a USEROPEN procedure address was specified then call the procedure
      ! to do the $OPEN and $CONNECT; it will return an RMS status code as
      ! procedure value. Otherwise do the $OPEN and $CONNECT ourselves.
      ! Set useropen flag, just as a debugging aid in case we get a dump with an SPR.
      !-
      IF .OPEN_ADR [OPEN$K_USEROPEN] NEQA 0
      THEN
      BEGIN
      LOCAL
      LOG_UNIT;
      ! Logical unit number
      LOG_UNIT = .CCB [LUB$W_LUN];
      ! Get the unit number
      CCB [LUB$V_USEROPEN] = 1;
      ! so we know the user opened the file!
      OPEN_STATUS = (.OPEN_ADR [OPEN$K_USEROPEN]) (FAB [0,0,0,0],
      .CCB, LOG_UNIT);
      END
      ELSE
      BEGIN
      ! not USEROPEN
      !+
      ! If old file is explicitly wanted, do an $OPEN. Otherwise
      ! (NEW, SCRATCH, UNKNOWN, default = NEW) do a $CREATE.
      ! UNKNOWN has set RMS FAB$V CIF to do an OPEN if file
      ! exists rather than a $CREATE. If file already existed
      ! on $CREATE (TYPE='UNKNOWN'), set LUB$V_OLD_FILE
      ! as flag that file already existed for error checking below.
      !-
      OPEN_STATUS = (
      IF .CCB [LUB$V_OLD_FILE]
      THEN
      $OPEN (FAB = FAB [0,0,0,0])
      ELSE
      $CREATE (FAB = FAB [0,0,0,0]);
      !+
      ! If no error in open/create, do $CONNECT (pointer to FAB already set in RAB).
      !-
      IF .OPEN_STATUS THEN OPEN_STATUS = $CONNECT (RAB = .CCB);
      END;
      !<BLF/PAGE>
```

```
1401 1447 2 !
1402 1448 2
1403 1449 2
1404 1450 2 Zero the XAB pointer in the FAB so we don't accidentally use it later.
1405 1451 2
1406 1452 2
1407 1453 2 FAB [FAB$L_XAB] = 0;
1408 1454 2
1409 1455 2
1410 1456 2 TYPE = 'UNKNOWN' has set RMS FAB$V_CIF to do an open if file exists
1411 1457 2 rather than a create. If file already existed on $CREATE (TYPE='UNKNOWN'),
1412 1458 2 set LUB$V_OLD_FILE as flag that file already existed for error checking below.
1413 1459 2
1414 1460 2
1415 1461 2 IF .FAB [FAB$V_CIF] AND .FAB [FAB$L_STS] NEQU RMS$_CREATED THEN CCB [LUB$V_OLD_FILE] = 1;
1416 1462 2
1417 1463 2
1418 1464 2 If CALL ASSIGN allocated space for the filename, deallocate it.
1419 1465 2
1420 1466 2
1421 1467 2 IF TESTBITSC (CCB [LUB$V_VIRT_RSN])
1422 1468 2 THEN
1423 1469 2     FOR$FREE_VM (.CCB [LUB$B_RSL], .CCB [LUB$A_RSN]);
1424 1470 2
1425 1471 2
1426 1472 2 If we have an expanded name string (or even better, a resultant name string),
1427 1473 2 point the LUB to it instead of the user supplied name. This will be
1428 1474 2 the file name used for error messages from now on.
1429 1475 2
1430 1476 2
1431 1477 2 IF .NAM [NAM$B_RSL] NEQ 0
1432 1478 2 THEN
1433 1479 2     BEGIN
1434 1480 2         CCB [LUB$A_RSN] = .NAM [NAM$L_RSA];
1435 1481 2         CCB [LUB$B_RSL] = .NAM [NAM$B_RSL];
1436 1482 2     END
1437 1483 2 ELSE
1438 1484 2
1439 1485 2     IF .NAM [NAM$B_ESL] NEQ 0
1440 1486 2     THEN
1441 1487 2         BEGIN
1442 1488 2             CCB [LUB$A_RSN] = .NAM [NAM$L_ESA];
1443 1489 2             CCB [LUB$B_RSL] = .NAM [NAM$B_ESL];
1444 1490 2         END;
1445 1491 2
1446 1492 2
1447 1493 2 !<BLF/PAGE>
```

```
1449      1494      2
1450      1495      2
1451      1496      2
1452      1497      2
1453      1498      2
1454      1499      2
1455      1500      2
1456      1501      2
1457      1502      2
1458      1503      2
1459      1504      2
1460      1505      2
1461      1506      2
1462      1507      2
1463      1508      2
1464      1509      2
1465      P 1510      2
1466      P 1511      2
1467      P 1512      2
1468      P 1513      2
1469      P 1514      2
1470      P 1515      2
1471      P 1516      2
1472      P 1517      2
1473      P 1518      2
1474      P 1519      2
1475      P 1520      2
1476      P 1521      2
1477      P 1522      2
1478      P 1523      2
1479      P 1524      2
1480      P 1525      2
1481      P 1526      2
1482      P 1527      2
1483      P 1528      2
1484      P 1529      2
1485      P 1530      2
1486      P 1531      2
1487      P 1532      2
1488      P 1533      2
1489      P 1534      2
1490      P 1535      2
1491      P 1536      2
1492      P 1537      2
1493      P 1538      2
1494      P 1539      2
1495      P 1540      2
1496      P 1541      2
1497      P 1542      2
1498      P 1543      2
1499      P 1544      2
1500      P 1545      2
1501      P 1546      2
1502      P 1547      2
1503      P 1548      2
1504      P 1549      2
1505      P 1550      2

      !
      ! If OPEN or CREATE error, SIGNAL STOP one of:
      ! FOR$_FILNOTFOU (29='FILE NOT FOUND') or
      ! FOR$_OPEFAI (30='OPEN FAILURE')
      ! FOR$_INCRELEN (37='INCONSISTENT RECORD LENGTH')
      ! FOR$_NO_SUCDEV (42='NO SUCH DEVICE')
      ! FOR$_FILNAMSPE (43='FILE NAME SPECIFICATION ERROR')
      ! FOR$_INVKEYSPE (49='INVALID KEY SPECIFICATION')
      ! Note: OPEN_STATUS can be anything for USEROPEN, so use status in FAB.
      !
      IF NOT .OPEN_STATUS
      THEN
        $FOR$$$SIGNAL_STO (
          (SELECTONEU .FAB [FAB$L_STS] OF
            SET
              [RMS$ FNF] :
                FOR$K_FILNOTFOU;          ! FILE NOT FOUND
              [RMS$ DEV] :
                FOR$K_NO_SUCDEV;          ! NO SUCH DEVICE
              [RMS$ FNM, RMS$ NOD, RMS$ TYP, RMS$ VER, RMS$ SYN] :
                FOR$K_FILNAMSPE;          ! FILE NAME SPECIFICATION ERROR
              [RMS$ POS, RMS$ SIZ, RMS$ NPK] :
                FOR$K_INVKEYSPE;          ! INVALID KEY SPECIFICATION
              [RMS$_CRE]:
                !
                ! Check for the special case of a mag tape file with
                ! blocksize less than recordsize (+ 4 if variable).
                ! If so, signal INCRELEN, since RMS does not give a
                ! useful message in this case; otherwise OPEFAI.
                !
              BEGIN
                LOCAL
                  OLD_STS,      ! Previous FAB$L_STS
                  OLD_STV;      ! Previous STV
                OLD_STS = .FAB [FAB$L_STS];
                OLD_STV = .FAB [FAB$L_STV];
                IF $PARSE (FAB = FAB [0,0,0,0])      ! Get device characteristics
                THEN
                  BEGIN
                    FAB [FAB$L_STS] = .OLD_STS;
                    FAB [FAB$L_STV] = .OLD_STV;
                    IF .BLOCK [FAB [FAB$L_DEV], DEV$V SOD; 1, LONG] AND .FAB [FAB$W_BLS] NEQ 0
                      ! If mag tape,
                    THEN
                      IF .FAB [FAB$W_BLS] LSSU .CCB [LUB$W_RBUF_SIZE]
```

```
.. 1506      P 1551      2
.. 1507      P 1552      2
.. 1508      P 1553      2
.. 1509      P 1554      2
.. 1510      P 1555      2
.. 1511      P 1556      2
.. 1512      P 1557      2
.. 1513      P 1558      2
.. 1514      P 1559      2
.. 1515      P 1560      2
.. 1516      P 1561      2
.. 1517      P 1562      2
.. 1518      P 1563      2
.. 1519      P 1564      2
.. 1520      P 1565      2
.. 1521      P 1566      2
.. 1522      P 1567      2
.. 1523      P 1568      2 !<BLF/PAGE>
```

```
      * (IF NOT .CCB [LUB$V_FIXED] THEN 4 ELSE 0)
      THEN
      FOR$K_INCRECLEN ! INCONSISTENT RECORD LENGTH
      ELSE
      FOR$K_OPEFAI ! OPEN FAILURE
      ELSE
      FOR$K_OPEFAI
      END
      ELSE
      FOR$K_OPEFAI
      END;
[OTHERWISE]:
      FOR$K_OPEFAI;
      TES));
```

```

1525 1569
1526 1570
1527 1571
1528 1572
1529 1573
1530 1574
1531 1575
1532 1576
1533 1577
1534 1578
1535 1579
1536 1580
1537 1581
1538 1582
1539 1583
1540 1584
1541 1585
1542 1586
1543 1587
1544 1588
1545 1589
1546 1590
1547 1591
1548 1592
1549 1593
1550 1594
1551 1595
1552 1596
1553 1597
1554 1598
1555 1599
1556 1600
1557 1601
1558 1602
1559 1603
1560 1604
1561 1605
1562 1606
1563 1607
1564 1608
1565 1609
1566 1610
1567 1611
1568 1612
1569 1613
1570 1614
1571 1615
1572 1616
1573 1617
1574 1618
1575 1619
1576 1620
1577 1621
1578 1622
1579 1623
1580 1624
1581 1625

```

```

!
!+
!-
If the file we just opened was an existing file, perform a couple of
consistency checks.
!-
IF .CCB [LUBSV_OLD_FILE]
THEN
  BEGIN
    !+
    !-
    Organization check:
    If user program did not specify organization with this OPEN,
    use the attributes from the file. If the user program did specify,
    check that it agrees with the file.
    !-

    IF .OPEN_ADR [OPEN$K_ORGANIZA] NEQ 0
    THEN
      BEGIN
        LOCAL
          T;

        T = (CASE .OPEN_ADR [OPEN$K_ORGANIZA] FROM OPEN$K_ORG_SEQ TO OPEN$K_ORG_IDX OF
          SET
            [OPEN$K_ORG_SEQ] : FAB$C_SEQ;
            [OPEN$K_ORG_REL] : FAB$C_REL;
            [OPEN$K_ORG_IDX] : FAB$C_IDX;
            [OUTRANGE] :
              BEGIN
                $FOR$$$SIGNAL_STO (FOR$K_INVARGFOR);
              END;
          TES);

        IF .T NEQ .FAB [FAB$B_ORG] THEN $FOR$$$SIGNAL_STO (FOR$K_INCFILORG);

        END;

    !+
    !-
    If ACCESS='KEYED' was specified and the file is not indexed,
    signal an error.
    !-

    IF (.CCB [LUBSV_KEYED] AND .FAB [FAB$B_ORG] NEQ FAB$C_IDX) OR (.CCB [LUBSV_DIRECT] AND .FAB [
      FAB$B_ORG] EQL FAB$C_IDX)
    THEN
      $FOR$$$SIGNAL_STO (FOR$K_INCFILORG);

    !+
    !-
    If the file does not have sequential organization, then set LUB bit.
    !-

    IF (.FAB [FAB$B_ORG] NEQ FAB$C_SEQ) THEN CCB [LUBSV_NOTSEQORG] = 1;

!<BLF/PAGE>

```

```
1583 1626 !
1584 1627
1585 1628
1586 1629
1587 1630
1588 1631
1589 1632
1590 1633
1591 1634
1592 1635
1593 1636
1594 1637
1595 1638
1596 1639
1597 1640
1598 1641
1599 1642
1600 1643
1601 1644
1602 1645
1603 1646
1604 1647
1605 1648
1606 1649
1607 1650
1608 1651
1609 1652
1610 1653
1611 1654
1612 1655
1613 1656
1614 1657
1615 1658
1616 1659
1617 1660
1618 1661
1619 1662
1620 1663
1621 1664
1622 1665
1623 1666
1624 1667
1625 1668
1626 1669
1627 1670
1628 1671
1629 1672
1630 1673
1631 1674
1632 1675
1633 1676
1634 1677
1635 1678
1636 1679
1637 1680
1638 1681
1639 1682

!
Record type check:
If user-program did not specified record-type in this OPEN,
use the file attributes. If user-program did specify
this OPEN, check that it agrees with the file.

CASE .OPEN_ADR [OPENS$K_RECORDTY] FROM 0 TO OPENS$K_REC_STMLF OF
  SET
    [0] : ! User did not specify
    BEGIN
      CCB [LUB$V_FIXED] = 0; ! Clear previously set bits
      CCB [LUB$V_SEGMENTED] = 0;
      IF .FAB [FAB$B_RFM] EQL FAB$C_FIX
      THEN
        CCB [LUB$V_FIXED] = 1 ! Fixed
      ELSE
        BEGIN ! Variable
          IF .CCB [LUB$V_DIRECT] AND NOT .CCB [LUB$V_NOTSEQORG]
          THEN
            $FOR$$SIGNAL_STO (FOR$K_INCRECTYP);
          IF NOT .CCB [LUB$V_NOTSEQORG] AND .CCB [LUB$V_UNFORMAT] AND
            NOT .CCB [LUB$V_DIRECT] AND (.FAB [FAB$B_RFM] EQL FAB$C_VAR)
          THEN
            CCB [LUB$V_SEGMENTED] = 1;
          END;
        END;
      [OPENS$K_REC_FIX] :
        IF .FAB [FAB$B_RFM] NEQU FAB$C_FIX THEN $FOR$$SIGNAL_STO (FOR$K_INCRECTYP);
      [OPENS$K_REC_VAR] :
        IF .FAB [FAB$B_RFM] NEQU FAB$C_VAR AND .FAB [FAB$B_RFM] NEQU FAB$C_VFC
        THEN
          $FOR$$SIGNAL_STO (FOR$K_INCRECTYP);
      [OPENS$K_REC_SEGM] :
        IF (.FAB [FAB$B_RFM] NEQU FAB$C_VAR) OR .CCB [LUB$V_NOTSEQORG]
        THEN
          $FOR$$SIGNAL_STO (FOR$K_INCRECTYP);
      [OPENS$K_REC_STM] :
        IF .FAB [FAB$B_RFM] NEQU FAB$C_STM
        THEN
          $FOR$$SIGNAL_STO (FOR$K_INCRECTYP);
      [OPENS$K_REC_STMCR] :
        IF .FAB [FAB$B_RFM] NEQU FAB$C_STMCR
```

```
.. 1640      1683      THEN
.. 1641      1684      $FOR$$$IGNAL_STO (FOR$K_INCRECTYP);
.. 1642      1685
.. 1643      1686      [OPEN$K_REC_STMLF] :
.. 1644      1687
.. 1645      1688      IF .FAB [FAB$B_RFM] NEQU FAB$C_STMLF
.. 1646      1689      THEN
.. 1647      1690      $FOR$$$IGNAL_STO (FOR$K_INCRECTYP);
.. 1648      1691
.. 1649      1692      [OUTRANGE] :
.. 1650      1693      $FOR$$$IGNAL_STO (FOR$K_INVARGFOR);
.. 1651      1694      TES;
.. 1652      1695
.. 1653      1696      !+
.. 1654      1697      ! Set maximum record number from file.
.. 1655      1698      !-
.. 1656      1699
.. 1657      1700      IF .CCB [LUB$SL_REC_MAX] EQL 0
.. 1658      1701      THEN
.. 1659      1702      CCB [LUB$SL_REC_MAX] = .FAB [FAB$SL_MRN]
.. 1660      1703      ELSE
.. 1661      1704
.. 1662      1705      IF .FAB [FAB$SL_MRN] NEQ 0 THEN CCB [LUB$SL_REC_MAX] = MIN (.CCB [LUB$SL_REC_MAX], .FAB [FAB$SL_MRN])
.. 1663      1706
.. 1664      1707
.. 1665      1708      !<BLF/PAGE>
```

```
1667 1709 3 !
1668 1710 3
1669 1711 3
1670 1712 3
1671 1713 3
1672 1714 3
1673 1715 3
1674 1716 3
1675 1717 3
1676 1718 3
1677 1719 3
1678 1720 3
1679 1721 3
1680 1722 3
1681 1723 3
1682 1724 3
1683 1725 4
1684 1726 3
1685 1727 3
1686 1728 3
1687 1729 3
1688 1730 3
1689 1731 4
1690 1732 4
1691 1733 3
1692 1734 3
1693 1735 3
1694 1736 3
1695 1737 4
1696 1738 4
1697 1739 3
1698 1740 3
1699 1741 3
1700 1742 3
1701 1743 3
1702 1744 4
1703 1745 3
1704 1746 3
1705 1747 3
1706 1748 3
1707 1749 3
1708 1750 3
1709 1751 3
1710 1752 3
1711 1753 4
1712 1754 4
1713 1755 4
1714 1756 4
1715 1757 4
1716 1758 3
1717 1759 3
1718 1760 3
1719 1761 3
1720 1762 3
1721 1763 3
1722 1764 3
1723 1765 4

+ Record size check:
  If user specified a record size (with DEFINE FILE or RECORDSIZE
  OPEN keyword, and MRS was required by RMS (fixed or relative),
  or organization indexed and MRS is non-zero, then they must agree.
  The recordsize the OTS will use is then computed in a reasonable
  manner.

+ If not a disk or terminal, use the blocksize as the maximum recordsize
  (if not there already).
IF (NOT .BLOCK [FAB [FAB$L_DEV], DEV$V_RND;4, BYTE]) AND
  (NOT .BLOCK [FAB [FAB$L_DEV], DEV$V_TRM;4, BYTE])
THEN
  IF .FAB [FAB$W_MRS] EQL 0
  THEN
    FAB [FAB$W_MRS] = .FAB [FAB$W_BLS];

  IF NOT .V_DEFAULT_SIZE AND (.CCB [LUB$V_FIXED]
    OR .FAB [FAB$B_ORG] EQL FAB$C_REL)
  THEN
    IF .CCB [LUB$W_RBUF_SIZE] NEQU .FAB [FAB$W_MRS] THEN $FOR$$SIGNAL_STO (FOR$K_INCRECLEN);

  IF (.CCB [LUB$V_FIXED]
    OR .FAB [FAB$B_ORG] EQL FAB$C_REL)
  THEN
    CCB [LUB$W_RBUF_SIZE] = .FAB [FAB$W_MRS]
  ELSE
    CCB [LUB$W_RBUF_SIZE] = MAXU (.CCB [LUB$W_RBUF_SIZE], .FAB [FAB$W_MRS], .XAB_BLOCK [XAB$W_LRL]);

  IF (.FAB [FAB$B_ORG] EQLU FAB$C_IDX) AND (NOT .CCB [LUB$V_FIXED])
  THEN
+ For variable indexed files, determine if the MRS is zero. If it is, this is an ISAM file
  created prior to FORTRAN V3 and should not be checked for buffer size agreement.
  If no explicit RECL was specified, use the bucket size to compute the buffersize.
  IF .FAB [FAB$W_MRS] EQLU 0
  THEN
    BEGIN
      IF .V_DEFAULT_SIZE
      THEN
        CCB [LUB$W_RBUF_SIZE] = .FAB [FAB$B_BKS] * 512;
      END
    ELSE
+ This is a new ISAM file. Check to be sure that the buffer size requested does
  not exceed the Max Recordsize specified when the file was created. Set the
  buffer size to the MRS to allow the records to grow.
  IF NOT .V_DEFAULT_SIZE AND
    (.CCB [LUB$W_RBUF_SIZE] GTRU .FAB [FAB$W_MRS])
```

```
1724      1766      3      THEN
1725      1767      4      $FOR$$SIGNAL_STO (FOR$K_INCRECLEN)
1726      1768      5      ELSE
1727      1769      6      CCB [LUB$W_RBUF_SIZE] = .FAB [FAB$W_MRS];
1728      1770      7
1729      1771      8      Key definition check. If file is ORGANIZATION='INDEXED' and
1730      1772      9      user specified a KEY definition, make sure it agrees with
1731      1773      A      what the file actually has. Key sizes must match, and key
1732      1774      B      datatypes must
1733      1775      C      match. If not, signal error FOR$ INVKEYSPE.
1734      1776      D      Make sure that we don't interfere with key XAB's that a
1735      1777      E      USEROPEN might have defined.
1736      1778      F
1737      1779      G
1738      1780      H      IF .FAB [FAB$B_ORG] EQL FAB$C_IDX
1739      1781      I      THEN
1740      1782      J      BEGIN          ! Indexed file
1741      1783      K
1742      1784      L      LOCAL
1743      1785      M      XAB_STATUS,          ! Status while freeing XABs
1744      1786      N      KEY_COUNT;          ! Count of OPEN defined keys
1745      1787      O
1746      1788      P      BEGIN
1747      1789      Q
1748      1790      R      LOCAL
1749      1791      S      KEY_DEFN : REF BLOCK [12, BYTE];
1750      1792      T
1751      1793      U      KEY_DEFN = .OPEN_ADR [OPEN$K_KEY];
1752      1794      V
1753      1795      W      IF .KEY_DEFN NEQ 0 THEN KEY_COUNT = .KEY_DEFN [OPEN$W_INFO] ELSE KEY_COUNT = 0;
1754      1796      X
1755      1797      Y      END;
1756      1798      Z
1757      1800      AA     XAB_STATUS=SS$ NORMAL;
1758      1801      AB     KEY_XAB = .XAB_BLOCK [XAB$L_NXT];
1759      1802      AC
1760      1803      AD     WHILE .KEY_XAB NEQU 0 AND .KEY_COUNT GTR 0 DO
1761      1804      AE     BEGIN          ! Go through XABs
1762      1805      AF
1763      1806      AG     IF (.KEY_XAB [XAB$B_COD] EQL XAB$C_KEY)
1764      1807      AH     THEN
1765      1808      AI     BEGIN
1766      1809      AJ     IF (.KEY_XAB [XAB$W_POS0] NEQ .KEY_XAB [OPEN$W_POS0]) OR (.KEY_XAB [XAB$B_SIZE] NEQ
1767      1810      AK     .KEY_XAB [OPEN$B_SIZE])
1768      1811      AL     THEN
1769      1812      AM     XAB_STATUS = FOR$K_INVKEYSPE;
1770      1813      AN
1771      1814      AO     IF .KEY_XAB [OPEN$B_KTYPE] NEQ .KEY_XAB [XAB$B_DTP]
1772      1815      AP     THEN
1773      1816      AQ     XAB_STATUS = FOR$K_INVKEYSPE;
1774      1817      AR
1775      1818      AS     BEGIN
1776      1819      AT
1777      1820      AU     LOCAL
1778      1821      AV     NEXT;          ! Address of next XAB in link
1779      1822      AW
```

```

: 1781      1823  7      NEXT = .KEY_XAB [XAB$L NXT];
: 1782      1824  7      FOR$FREE_VM (OPEN$K_XAB_SIZE, .KEY_XAB);
: 1783      1825  7      KEY_XAB = .NEXT;
: 1784      1826  6      END;
: 1785      1827  6      KEY_COUNT = .KEY_COUNT - 3;
: 1786      1828  5      END;
: 1787      1829  5
: 1788      1830  4      END;      ! Go through XABs
: 1789      1831  4
: 1790      1832  4      +
: 1791      1833  4      | If we had discovered any error while freeing the XAB's
: 1792      1834  4      | we report it now.  If we had reported it when we found it,
: 1793      1835  4      | we would have been left with some XABs laying around
: 1794      1836  4      | whose memory had not been deallocated.
: 1795      1837  4      | -
: 1796      1838  4
: 1797      1839  4      IF NOT .XAB_STATUS
: 1798      1840  4      THEN
: 1799      1841  4          $FOR$SIGNAL_STO (.XAB_STATUS);
: 1800      1842  4
: 1801      1843  3      END;      ! Indexed file
: 1802      1844  3
: 1803      1845  3      END
: 1804      1846  2      ELSE
: 1805      1847  2      !<BLF/PAGE>      ! End of old file processing
```

```
1807 1848 2
1808 1849 2
1809 1850 2
1810 1851 2
1811 1852 2
1812 1853 2
1813 1854 2
1814 1855 2
1815 1856 2
1816 1857 2
1817 1858 2
1818 1859 2
1819 1860 2
1820 1861 2
1821 1862 2
1822 1863 2
1823 1864 2
1824 1865 2
1825 1866 2
1826 1867 2
1827 1868 2
1828 1869 2
1829 1870 2
1830 1871 2
1831 1872 2
1832 1873 2
1833 1874 2
1834 1875 2
1835 1876 2
1836 1877 2
1837 1878 2
1838 1879 2
1839 1880 2
1840 1881 2
1841 1882 2
1842 1883 2
1843 1884 2
1844 1885 2
1845 1886 2
1846 1887 2
1847 1888 2
1848 1889 2
1849 1890 2
1850 1891 2
1851 1892 2
1852 1893 2
1853 1894 2
1854 1895 2
1855 1896 2
1856 1897 2
1857 1898 2
1858 1899 2
1859 1900 2
1860 1901 2
1861 1902 2
1862 1903 2
1863 1904 2

!
+ Else (file was created)
- Make sure V_APPEND is off so BACKSPACE will work.

      BEGIN
      CCB [LUB$V_APPEND] = 0;
      END;

+ If this is not a disk or terminal, and if RECL was not specified,
- then reduce the default recordsize to fit within the blocksize.

IF .V_DEFAULT_SIZE
THEN
  IF (NOT .BLOCK [FAB [FAB$L_DEV], DEV$V_RND;4, BYTE]) AND
  (NOT .BLOCK [FAB [FAB$L_DEV], DEV$V_TRM;4, BYTE]) AND
  (.FAB [FAB$W_BLS] NEQ 0)
  THEN
    BEGIN
    LOCAL
    NEW RECL: WORD;
    NEW RECL = .FAB [FAB$W_BLS];
    IF .CCB [LUB$V_SEGMENTED]
    THEN
      NEW RECL = .NEW RECL - 4; ! Compensate for length
    IF .NEW RECL LSSU .CCB [LUB$W_RBUF_SIZE]
    THEN
      CCB [LUB$W_RBUF_SIZE] = .NEW RECL;
    END;

+ If this is a process-permanent file, ignore the carriage-control
- attributes RMS returned in the FAB and use the ones we set
originally. RMS will properly convert our writes anyway.

IF .NAM [NAM$V_PPF]
THEN
  FAB [FAB$B_RAT] = .ORIG_RAT;

+ Set up the list-directed output record size as RECL, if specified,
- else 81 (80 if not FORTRAN carriage control).

CCB [LUB$W_R_MARGIN] = (IF NOT .V_DEFAULT_SIZE THEN .CCB [LUB$W_RBUF_SIZE] ELSE
  (IF .FAB [FAB$V_FTN] THEN 81 ELSE 80));

+ Set bits in the LUB to indicate the file's carriage control
- characteristics. This information is used by INQUIRE.
```

```
1864 1905 2 IF .FAB [FAB$V_FTN]
1865 1906 THEN
1866 1907   CCB [LUB$V_FTN] = 1;
1867 1908 IF .FAB [FAB$V_CR]
1868 1909 THEN
1869 1910   CCB [LUB$V_CR] = 1;
1870 1911 IF .FAB [FAB$V_PRN]
1871 1912 THEN
1872 1913   CCB [LUB$V_PRN] = 1;
1873 1914
1874 1915   !+
1875 1916   ! Allocate record buffer dynamically from LUB$W_RBUF_SIZE setting in bytes.
1876 1917   ! Set LUB$A_RBUF_ADR to address of buffer allocated.
1877 1918   !-
1878 1919
1879 1920   CCB [LUB$A_RBUF_ADR] = FOR$$GET_VM (.CCB [LUB$W_RBUF_SIZE]);
1880 1921
1881 1922   !+
1882 1923   ! Allocate dynamic storage for the file name so the name can be
1883 1924   ! used later on for error diagnostics. Point the LUB to the new
1884 1925   ! location. (The size is already correct!)
1885 1926   ! Indicate that the string name is now stored in virtual memory so
1886 1927   ! it will be deallocated!
1887 1928   !-
1888 1929
1889 1930 BEGIN
1890 1931
1891 1932 LOCAL
1892 1933   T;
1893 1934
1894 1935   T = FOR$$GET_VM (.CCB [LUB$B_RSL]);
1895 1936   CH$MOVE (.CCB [LUB$B_RSL], .CCB [LUB$A_RSN], .T);
1896 1937   CCB [LUB$A_RSN] = .T;
1897 1938   NAM [NAM$B_RSA] = .T;
1898 1939   NAM [NAM$B_ESA] = .T;
1899 1940   NAM [NAM$B_ESL] = .CCB [LUB$B_RSL];
1900 1941   CCB [LUB$V_VIRT_RSN] = 1;
1901 1942 END;
1902 1943
1903 1944   !+
1904 1945   ! Store a code in the LUB indicating the type of organization.
1905 1946   !-
1906 1947
1907 1948 SELECTONE (.FAB [FAB$B_ORG]) OF
1908 1949   SET
1909 1950
1910 1951   [FAB$C_SEQ] :
1911 1952     CCB [LUB$B_ORGAN] = LUB$K_ORG_SEQUE;
1912 1953
1913 1954   [FAB$C_REL] :
1914 1955     CCB [LUB$B_ORGAN] = LUB$K_ORG_RELAT;
1915 1956
1916 1957   [FAB$C_IDX] :
1917 1958     BEGIN
1918 1959
1919 1960     IF .CCB [LUB$V_SEGMENTED] THEN $FOR$$SIGNAL_STO (FOR$K_INCRECTYP);
1920 1961
```

```
1921      CCB [LUB$B_ORGAN] = LUB$K_ORG_INDEX;  
1922      END;  
1923  
1924      [OTHERWISE] :  
1925      $FOR$$SIGNAL_STO (FOR$K_INCFILORG);  
1926      TES;  
1927  
1928      !+  
1929      ! Set RAB fields that seldom change: UBF and USZ  
1930      !-  
1931  
1932      CCB [RAB$L_UBF] = .CCB [LUB$A_RBUF_ADR];  
1933      CCB [RAB$W_USZ] = .CCB [LUB$W_RBUF_SIZE];  
1934      CCB [LUB$A_UBF] = .CCB [LUB$A_RBUF_ADR];  
1935  
1936      !+  
1937      ! If the file is a sequential organization, sequential access,  
1938      ! disk file which is not a PPF, enable RFA cacheing for BACKSPACE.  
1939      !-  
1940  
1941      IF NOT .CCB [LUB$V_NOTSEQORG] AND  
1942      NOT .CCB [LUB$V_DIRECT] AND  
1943      NOT .CCB [LUB$V_FIXED] AND  
1944      NOT .NAM [NAM$V_PPF] AND  
1945      NOT .FAB [FAB$V_SQO]  
1946      THEN  
1947      BEGIN  
1948      BIND  
1949      FAB_DEV = FAB [FAB$L_DEV]: BLOCK [4, BYTE];  
1950      IF .FAB_DEV [DEV$V_RND] ? Random-access device?  
1951      THEN  
1952      BEGIN  
1953      LOCAL  
1954      RCE: REF RCE_R_RCE_STRUCT,  
1955      OLD_RCE: REF RCE_R_RCE_STRUCT;  
1956  
1957      !+  
1958      ! Allocate space for the RFA cache entries.  
1959      !-  
1960  
1961      RCE = FOR$$GET VM (  
1962      (RCE_K_CACHE_SIZE * RCE_S_RCE_STRUCT));  
1963  
1964      !+  
1965      ! Create a circularly linked list of entries and zero the  
1966      ! LOG_RECNO field of each entry.  
1967      !-  
1968  
1969      CCB [LUB$A_RFA_CACHE_BEG] = .RCE;      ! First allocated byte  
1970      CCB [LUB$A_RFA_CACHE_PTR] = .RCE;      ! Current entry  
1971      OLD_RCE = .RCE + (RCE_K_CACHE_SIZE - 1) * RCE_S_RCE_STRUCT;  
1972      DECRU I FROM RCE_K_CACHE_SIZE TO 1 DO  
1973      BEGIN  
1974      OLD_RCE [RCE_A_NEXT] = .RCE;  
1975      RCE [RCE_A_PREV] = .OLD_RCE;  
1976      RCE [RCE_L_LOG_RECNO] = 0;  
1977      OLD_RCE = .RCE;
```

```
1978 2019 5 RCE = .RCE + RCE_S_RCE_STRUCT;
1979 2020 4 END;
1980 2021 4
1981 2022 4 CCB [LUB$V_RFA_CACHE_ENABLE] = 1;
1982 2023 3 END;
1983 2024 2 END;
1984 2025 2
1985 2026 2
1986 2027 2 + Indicate that the file is now FORTRAN opened.
1987 2028 2 -
1988 2029 2 CCB [LUB$B_LANGUAGE] = LUB$K_LANG_FOR;
1989 2030 2 CCB [LUB$V_OPENED] = 1;
1990 2031 2
1991 2032 2 + Make sure that the FORTRAN exit handler will be called when the image
1992 2033 2 exits to purge the file's I/O buffers and close it, if necessary.
1993 2034 2 -
1994 2035 2
1995 2036 2 IF ( NOT .FOR$SL_XIT_LOCK ) THEN FOR$DECL_EXITH ();
1996 2037 2
1997 2038 2 RETURN; ! Return from OPEN_PROC routine
1998 2039 1 END; ! End of OPEN_PROC routine
```

54	41	44	2E	54	50	45	43	43	41	52	4F	46	00045	P.AAD:	.ASCII	\FOR\$READ:\		
		3A	54	50	45	43	43	41	24	52	4F	46	0005B	P.AAF:	.ASCII	\FOR\$ACCEPT:\		
		54	41	44	2E	45	50	59	54	52	4F	46	00066	P.AAG:	.ASCII	\FORTYPE.DAT\		
				3A	45	50	59	54	24	52	4F	46	00071	P.AAH:	.ASCII	\FOR\$TYPE:\		
		54	41	44	2E	54	4E	49	52	50	52	4F	46	0007A	P.AAI:	.ASCII	\FORPRINT.DAT\	
				3A	54	4E	49	52	50	24	52	4F	46	00086	P.AAJ:	.ASCII	\FOR\$PRINT:\	

										A_SYSS\$INPUT=	P.AAA	
										A_SYSS\$OUTPUT=	P.AAB	
										.EXTRN	SYS\$TRNLOG,	SYS\$PARSE
										.EXTRN	SYS\$OPEN,	SYS\$CREATE
										.EXTRN	SYS\$CONNECT	

										07FC	00000	.ENTRY	FOR\$OPEN_PROC, Save R2,R3,R4,R5,R6,R7,R8,-	0291				
													R9,R10					
										5E	FD88	CE	9E	00002	MOVAB	-632(SP), SP		
											E8	AB	D5	00007	TSTL	-24(CCB)	0406	
												3C	13	0000A	BEQL	1\$		
										56	E8	AB	D0	0000C	MOVL	-24(CCB), HEAP_FAB	0411	
										50	01	A6	9A	00010	MOVZBL	1(HEAP_FAB), R0	0412	
										44	AB				MOVCL	R0, (HEAP_FAB), 68(CCB)		
										66		50	28	00014	PUSHL	HEAP_FAB	0413	
												56	DD	00019	MOVZBL	1(HEAP_FAB), -(SP)		
										7E	01	A6	9A	0001B	CALLS	#2, FOR\$FREE_VM		
										00000000G	00		02	FB	0001F	CLRL	-24(CCB)	0414
												E8	AB	D4	00026	MOVZBL	120(CCB), R6	0415
										56	78	AB	9A	00029	BEQL	1\$		
												19	13	0002D	MOVCL	R6, @112(CCB), TEMP_FNS	0418	
										FEC8	CD	70	BB		PUSHL	112(CCB)	0419	
												70	AB	DD	00036			

		00000000G	00		56	DD	00039		PUSHL	R6		
		70	AB	FEC8	02	FB	0003B		CALLS	#2, FOR\$FREE VM		
			59	0094	CD	9E	00042	1\$:	MOVAB	TEMP_FNS, 112(CCB)	0420	
		6C	AB		CB	9E	00048		MOVAB	148(R11), R9	0428	
		24	AE	0098	59	D0	0004D		MOVL	R9, 108(CCB)		
		20	AE	00A0	CB	9E	00051		MOVAB	152(CCB), 36(SP)	0429	
			50	40	CB	9E	00057		MOVAB	160(CCB), 32(SP)		
		20	BE		AE	9E	0005D		MOVAB	RES_OR_EXP_NAME, R0		
		24	BE		50	D0	00061		MOVL	R0, @32(SP)		
		009E	CB		50	D0	00065		MOVL	R0, @36(SP)		
		0096	CB		01	8E	00069		MNEGB	#1, 158(CCB)	0430	
			6E		01	8E	0006E		MNEGB	#1, 150(CCB)		
2C	00				00	2C	00073		MCVCS	#0, (SP), #0, #44, \$RMS_PTR	0431	
					AD		00078					
		C8	AD	2C1D	8F	B0	0007A		MOVW	#11293, \$RMS_PTR		
		68	AB		AD	9E	00080		MOVAB	XAB_BLOCK, 104(CCB)	0432	
			58		AD	9E	00085		MOVAB	XAB_BLOCK, KEY XAB	0433	
		0C	AE		AD	9E	00089		MOVAB	72(CCB), 12(SP)	0440	
		0C	BE		AB	9E	0008E		BISB2	#32, @12(SP)		
					20	88	0008E		CLRL	A DEF LOGNAM	0469	
			56		55	D4	00092		MOVAB	121(CCB), R6	0544	
			52		AB	9E	00094		MOVAB	116(CCB), R2	0545	
			53		AB	9E	00098		MOVAB	112(CCB), R3	0580	
	03	FFFC	8F		AB	9E	0009C		MOVAB	-58(CCB), #4, #3	0471	
0103	00F0	00D5			C6	AF	000A0	2\$:	CASEW	10\$-2\$,-		
							000A7		.WORD	11\$-2\$,-		
										13\$-2\$,-		
										14\$-2\$,-		
			50		AC	D0	000AF		MOVL	OPEN_ADR, R0	0516	
			57		A0	D0	000B3		MOVL	56(R0), R7		
					05	13	000B7		BEQL	3\$		
					68	A0	D5	000B9	TSTL	104(R0)	0517	
					56	12	000BC		BNEQ	4\$		
		F4	AD	4F46	8F	B0	000BE	3\$:	MOVW	#20294, T_DFLT_FILE_NAM	0520	
		F6	AD	52	8F	90	000C4		MOVW	#82, T_DFLT_FILE_NAM+2	0522	
			51	C6	AB	32	000C9		CVTWL	-58(CCB), R1	0523	
			51	00000064	8F	C6	000CD		DIVL2	#100, R1		
			51		01	7A	000D4		EMUL	#1, R1, #0, -(SP)		
7E	00		51		0A	7B	000D9		EDIV	#10, (SP)+, R1, R1		
51	F7	AD	51		30	81	000DE		ADDB3	#48, R1, T_DFLT_FILE_NAM+3		
			51		AB	32	000E3		CVTWL	-58(CCB), R1	0524	
			51		0A	C6	000E7		DIVL2	#10, R1		
			51		01	7A	000EA		EMUL	#1, R1, #0, -(SP)		
7E	00		51		0A	7B	000EF		EDIV	#10, (SP)+, R1, R1		
51	F8	AD	51		30	81	000F4		ADDB3	#48, R1, T_DFLT_FILE_NAM+4		
			51		AB	32	000F9		CVTWL	-58(CCB), R1	0525	
			51		01	7A	000FD		EMUL	#1, R1, #0, -(SP)		
7E	00		51		0A	7B	00102		EDIV	#10, (SP)+, R1, R1		
51	F9	AD	51		30	81	00107		ADDB3	#48, R1, T_DFLT_FILE_NAM+5		
					8F	D0	0010C		MOVL	#1413563438, T_DFLT_FILE_NAM+6	0526	
		FA	AD	5441442E	A0	D5	00114	4\$:	TSTL	104(R0)	0537	
					14	13	00117		BEQL	5\$		
			54		A0	D0	00119		MOVL	104(R0), NAM_DSC	0542	
		00FF	8F		64	B1	0011D		CMPL	(NAM_DSC), #255	0543	
					1C	1A	00122		BGTRU	7\$		
			66		64	90	00124		MOVW	(NAM_DSC), (R6)	0544	
			62		A4	D0	00127		MOVL	4(NAM_DSC), (R2)	0545	

		07	11	0012B	BRB	6\$	0537	
	66	0A	90	0012D	5\$:	MOVAB	#10, (R6)	0554
	62	F4	AD	9E 00130		MOVAB	T_DFLT_FILE_NAM, (R2)	0555
			57	D5 00134	6\$:	TSTL	R7	0563
			17	13 00136		BEQL	9\$	
	52		57	D0 00138		MOVL	R7, NAM_DSC	0575
00FF	8F		62	B1 0013B		CMPW	(NAM_DSC), #255	0577
			03	1B 00140	7\$:	BLEQU	8\$	
		06AD	31	00142		BRW	139\$	
78	AB		62	90 00145	8\$:	MOVAB	(NAM_DSC), 120(CCB)	0579
	63	04	A2	D0 00149		MOVL	4(NAM_DSC), (R3)	0580
			74	11 0014D		BRB	16\$	0563
			63	D5 0014F	9\$:	TSTL	(R3)	0592
			70	12 00151		BNEQ	16\$	
78	AB		06	90 00153		MOVAB	#6, 120(CCB)	0595
	63	F4	AD	9E 00157		MOVAB	T_DFLT_FILE_NAM, (R3)	0596
	05	C6	AB	B1 0015B		CMPW	-58(CCB), #5	0604
			2C	13 0015F		BEQL	12\$	
	06	C6	AB	B1 00161		CMPW	-58(CCB), #6	0612
			5C	12 00165		BNEQ	16\$	
			52	11 00167		BRB	15\$	0615
	66		0B	90 00169	10\$:	MOVAB	#11, (R6)	0476
	62	FE3A	CF	9E 0016C		MOVAB	P.AAC, (R2)	0477
78	AB		09	90 00171		MOVAB	#9, 120(CCB)	0478
	63	FE3C	CF	9E 00175		MOVAB	P.AAD, (R3)	0479
			11	11 0017A		BRB	12\$	0480
	66		0D	90 0017C	11\$:	MOVAB	#13, (R6)	0486
	62	FE3B	CF	9E 0017F		MOVAB	P.AAE, (R2)	0487
78	AB		0B	90 00184		MOVAB	#11, 120(CCB)	0488
	63	FE3F	CF	9E 00188		MOVAB	P.AAF, (R3)	0489
	55	FE04	CF	9E 0018D	12\$:	MOVAB	A.SYS\$INPUT, A.DEF_LOGNAM	0490
	54		0A	D0 00192		MOVL	#T0, L_DEF_LOGNAM	0491
			2C	11 00195		BRB	16\$	0471
	66		0B	90 00197	13\$:	MOVAB	#11, (R6)	0496
	62	FE38	CF	9E 0019A		MOVAB	P.AAG, (R2)	0497
78	AB		09	90 0019F		MOVAB	#9, 120(CCB)	0498
	63	FE3A	CF	9E 001A3		MOVAB	P.AAH, (R3)	0499
			11	11 001A8		BRB	15\$	0500
	66		0C	90 001AA	14\$:	MOVAB	#12, (R6)	0506
	62	FE39	CF	9E 001AD		MOVAB	P.AAI, (R2)	0507
78	AB		0A	90 001B2		MOVAB	#10, 120(CCB)	0508
	63	FE3C	CF	9E 001B6		MOVAB	P.AAJ, (R3)	0509
	55	FDE0	CF	9E 001BB	15\$:	MOVAB	A.SYS\$OUTPUT, A.DEF_LOGNAM	0510
	54		0B	D0 001C0		MOVL	#T1, L_DEF_LOGNAM	0511
			55	D5 001C3	16\$:	TSTL	A.DEF_LOGNAM	0631
			47	13 001C5		BEQL	18\$	
3A	AE	010E	8F	B0 001C7		MOVW	#270, LOGNAM_DSC+2	0640
30	AE	010E00FF	8F	D0 001CD		MOVL	#17694975, RESULT_DSC	0643
34	AE	40	AE	9E 001D5		MOVAB	RES OR EXP NAME, RESULT_DSC+4	0644
3C	AE		63	D0 001DA		MOVL	(R3), LOGNAM_DSC+4	0645
38	AE	78	AB	9B 001DE		MOVZBW	120(CCB), LOGNAM_DSC	0646
		C6	AB	B5 001E3		TSTW	-58(CCB)	0648
			03	18 001E6		BGEQ	17\$	
		38	AE	B7 001E8		DECW	LOGNAM_DSC	0655
			7E	7C 001EB	17\$:	CLRQ	-(SP)	0664
			7E	D4 001ED		CLRL	-(SP)	
		3C	AE	9F 001EF		PUSHAB	RESULT_DSC	

00000000G	00	4C	7E	D4	001F2	CLRL	-(SP)	
00000629	8F		AE	9F	001F4	PUSHAB	LOGNAM DSC	
			06	FB	001F7	CALLS	#6, SY\$STRNLOG	
			50	D1	001FE	CMPL	R0, #1577	
			07	12	00205	BNEQ	18\$	
	63		55	D0	00207	MOVL	A_DEF_LOGNAM, (R3)	0667
78	AB		54	90	0020A	MOVB	L_DEF_LOGNAM, 120(CCB)	0668
14	AE	F8	AB	9E	0020E	MOVAB	-8(CCB), 20(SP)	0679
14	BE		63	D0	00213	MOVL	(R3), @20(SP)	
1C	AE	F7	AB	9E	00217	MOVAB	-9(CCB), 28(SP)	0680
1C	BE	78	AB	90	0021C	MOVB	120(CCB), @28(SP)	
		6C	AB	D4	00221	CLRL	108(CCB)	0688
		44	AB	9E	00224	MOVAB	68(CCB), 8(SP)	0689
08	AE	08	AE	DD	00229	PUSHL	8(SP)	
00000000G	00		01	FB	0022C	CALLS	#1, SY\$PARSE	
	0B		50	E9	00233	BLBC	R0, 19\$	
05	0085		05	E1	00236	BBC	#5, 133(CCB), 19\$	0694
0C	BE	40	8F	88	0023C	BISB2	#64, @12(SP)	0696
		4C	AB	D4	00241	CLRL	76(CCB)	0698
			59	D0	00244	MOVL	R9, 108(CCB)	0699
6C	AB	04	AC	D0	00248	MOVL	OPEN ADR, R7	0708
57		20	A7	E9	0024C	BLBC	32(R7), 20\$	
06			04	88	00250	BISB2	#4, -4(CCB)	0711
FC	AB		09	11	00254	BRB	21\$	0708
		5A	AB	95	00256	TSTB	90(CCB)	0715
			04	12	00259	BNEQ	21\$	
5A	AB		1F	90	0025B	MOVB	#31, 90(CCB)	0717
		E0	AB	D5	0025F	TSTL	-32(CCB)	0728
			04	12	00262	BNEQ	22\$	
E0	AB		01	D0	00264	MOVL	#1, -32(CCB)	0730
0C	BE	0400	8F	AB	00268	BISW2	#1024, @12(SP)	0732
	00	10	A7	CF	0026E	CASEL	16(R7), #0, #4	0734
002E	0027		0027		00273	.WORD	25\$-23\$,-	
	000C		0046		0027B		24\$-23\$,-	
							25\$-23\$,-	
							26\$-23\$,-	
							28\$-23\$,-	
			5A	11	0027D	BRB	31\$	0773
			10	88	0027F	BISB2	#16, -4(CCB)	0739
FC	AB	40	8F	8A	00283	BICB2	#64, @12(SP)	0740
0C	BE		01	90	00288	MOVB	#1, 30(CCB)	0741
1E	AB	E0	AB	9E	0028C	MOVAB	-32(CCB), 1\$ (CCB)	0742
30	AB	34	AB	94	00291	CLRB	52(CCB)	0743
			10	88	00294	BISB2	#16, 4(CCB)	0744
04	AB		30	11	00298	BRB	29\$	0734
		40	8F	88	0029A	BISB2	#64, -3(CCB)	0749
FD	AB		13	11	0029F	BRB	27\$	0750
			02	E0	002A1	BBS	#2, -4(CCB), 36\$	0755
55	FC		01	88	002A6	BISB2	#1, 5(CCB)	0758
	AB		20	88	002AA	BISB2	#32, -3(CCB)	0759
05	AB	0400	8F	AA	002AE	BICW2	#1024, @12(SP)	0760
FD	AB	1E	AB	94	002B4	CLRB	30(CCB)	0761
0C	BE		11	11	002B7	BRB	29\$	0734
		40	8F	8A	002B9	BICB2	#64, @12(SP)	0766
1E	AB		01	90	002BE	MOVB	#1, 30(CCB)	0767
		35	AB	94	002C2	CLRB	53(CCB)	0768
FD	AB	80	8F	88	002C5	BISB2	#128, -3(CCB)	0769

001A	04 0013	00 000D	3C	A7 0013 002F	CF	002CA 002CF 002D7	29\$: 30\$:	CASEL .WORD	60(R7) NO, #4 33\$-30\$,- 32\$-30\$,- 33\$-30\$,- 34\$-30\$,- 37\$-30\$		0783
				02EA	31	002D9	31\$:	BRW	101\$		0818
		FC	AB	08	88	002DC	32\$:	BISB2	#8, -4(CCB)		0787
	22	FC	AB	27	11	002E0		BRB	38\$		
				03	E0	002E2	33\$:	BBS	#3, -4(CCB), 38\$		0791
				08	11	002E7		BRB	35\$		0793
		FC	AB	20	88	002E9	34\$:	BISB2	#32, -4(CCB)		0801
		OC	BE	10	88	002ED		BISB2	#16, @12(SP)		0802
	05	FC	AB	02	E0	002F1	35\$:	BBS	#2, -4(CCB), 36\$		0803
	0E	FD	AB	05	E1	002F6		BBC	#5, -3(CCB), 38\$		0804
				01C5	31	002FB	36\$:	BRW	84\$		0806
OC		01	19	01	F0	002FE	37\$:	INSV	#1, #25, #1, @12(SP)		0811
BE		F2	AB	02	E0	00304		BBS	#2, -4(CCB), 36\$		0812
			52	08	A7	D0	00309	38\$:	MOVL	8(R7), R2	0831
			53		01	D0	0030D		MOVL	#1, R3	
					52	D5	00310		TSTL	R2	0834
					02	12	00312		BNEQ	39\$	
			01		53	D4	00314		CLRL	R3	
					52	D1	00316	39\$:	CMPL	R2, #1	0837
					07	12	00319		BNEQ	40\$	
					53	D4	0031B		CLRL	R3	
	D9	FC	AB	05	E0	0031D		BBS	#5, -4(CCB), 36\$		0839
			02	52	D1	00322	40\$:	CMPL	R2, #2		0841
				0A	13	00325		BEQL	41\$		
			05	52	D1	00327		CMPL	R2, #5		
				11	19	0032A		BLSS	42\$		
			06	52	D1	0032C		CMPL	R2, #6		
				OC	14	0032F		BGTR	42\$		
				53	D4	00331	41\$:	CLRL	R3		
	C3	FC	AB	02	E0	00333		BBS	#2, -4(CCB), 36\$		0844
		FC	AB	08F	88	00338		BISB2	#64, -4(CCB)		0847
			03	52	D1	0033D	42\$:	CMPL	R2, #3		0850
				05	13	00340		BEQL	43\$		
			05	52	D1	00342		CMPL	R2, #5		
				OC	12	00345		BNEQ	44\$		
				53	D4	00347	43\$:	CLRL	R3		
				05	E0	00349		BBS	#5, -4(CCB), 36\$		0854
	AD	FC	AB	8F	88	0034E		BISB2	#128, -4(CCB)		0856
		FC	AB	52	D1	00353	44\$:	CMPL	R2, #4		0859
			04	05	13	00356		BEQL	45\$		
				52	D1	00358		CMPL	R2, #6		
			06	0B	12	0035B		BNEQ	46\$		
				53	D4	0035D	45\$:	CLRL	R3		
				05	E0	0035F		BBS	#5, -4(CCB), 36\$		0863
	97	FC	AB	20	88	00364		BISB2	#32, -1(CCB)		0867
		FF	AB	53	E8	00368	46\$:	BLBS	R3, 53\$		0871
			3A	14	A7	CF	0036B		CASEL	20(R7), #-1, #3	0882
001A		03 FFFFFFFF	8F	001E		00374	47\$:	.WORD	51\$-47\$,- 48\$-47\$,- 49\$-47\$,- 50\$-47\$		
	0014	000A							64\$		0903
				71	11	0037C		BRB			

Address	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419	Op420	Op421	Op422	Op423	Op424	Op425	Op426	Op427	Op428	Op429	Op430	Op431	Op432	Op433	Op434	Op435	Op436	Op437	Op438	Op439	Op440	Op441	Op442	Op443	Op444	Op445	Op446	Op447	Op448	Op449	Op450	Op451	Op452	Op453	Op454	Op455	Op456	Op457	Op458	Op459	Op460	Op461	Op462	Op463	Op464	Op465	
---------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--

			05	18	0042E	BGEQ	73\$		
			4C	A7	D5 00430	TSTL	76(R7)		
				87	12 00433	BNEQ	57\$		
			61	AB	94 00435	CLRB	97(CCB)		1024
		01		08	8A 00438	BICB2	#8, 1(R9)		1025
				23	11 0043C	BRB	77\$		1010
15		FD	AB	03	E0 0043E	BBS	#3, -3(CCB), 76\$		1031
			FD	AB	95 00443	TSTB	-3(CCB)		
				7B	19 00446	BLSS	84\$		
		61	AB	10	90 00448	MOVB	#16, 97(CCB)		1033
				13	11 0044C	BRB	77\$		1010
70		FC	AB	04	E0 0044E	BBS	#4, -4(CCB), 84\$		1039
68		FD	AB	05	E0 00453	BBS	#5, -3(CCB), 84\$		
66		FD	AB	03	E0 00458	BBS	#3, -3(CCB), 84\$		
		61	AB	20	90 0045D	MOVB	#32, 97(CCB)		1043
08			69	0B	E1 00461	BBC	#11, (R9), 78\$		1054
50	0E000000		8F	63	AB	78 00465	ASHL	99(CCB), #234881024, R0	1055
				53	19 0046E	BLSS	84\$		
				50	A7	D5 00470	TSTL	80(R7)	1065
					2D	12 00473	BNEQ	81\$	
			10	61	AB	91 00475	CMPB	97(CCB), #16	1068
					10	13 00479	BEQL	79\$	
			20	61	AB	91 0047B	CMPB	97(CCB), #32	
					0A	13 0047F	BEQL	79\$	
05		FC	AB	04	E0 00481	BBS	#4, -4(CCB), 79\$		
			FD	AB	95 00486	TSTB	-3(CCB)		1069
				0A	18 00489	BGEQ	80\$		
		63	AB	01	90 0048B	MOVB	#1, 99(CCB)		1072
		FD	AB	04	88 0048F	BISB2	#4, -3(CCB)		1073
				0D	11 00493	BRB	81\$		1068
				02	90 00495	MOVB	#2, 99(CCB)		1077
04		FD	AB	01	E1 00499	BBC	#1, -3(CCB), 81\$		1079
		FD	AB	08	88 0049E	BISB2	#8, -3(CCB)		
		0D		34	A7	E9 004A2	BLBC	52(R7), 82\$	1089
		5B	AB		0F	90 004A6	MOVB	#15, 91(CCB)	1092
05			69		0B	E0 004AA	BBS	#11, (R9), 82\$	1094
		5B	AB	40	8F	88 004AE	BISB2	#64, 91(CCB)	1096
		18	AE	5C	A7	D0 004B3	MOVL	92(R7), 24(SP)	1107
					03	12 004B8	BNEQ	83\$	
				00FE	31 004BA	BRW	99\$		
			20	61	AB	91 004BD	CMPB	97(CCB), #32	1117
					05	13 004C1	BEQL	85\$	
					2E	DD 004C3	PUSHL	#46	
				065A	31 004C5	BRW	214\$		
			56	18	AE	D0 004C8	MOVL	24(SP), KEY_DEFN	1119
		04	AE	02	A6	32 004CC	CVTWL	2(KEY_DEFN), KEY_COUNT	1120
			56		04	C0 004D1	ADDL2	#4, KEY_DEFN	1121
7E		00	AE		01	7A 004D4	EMUL	#1, KEY_COUNT, #0, -(SP)	1123
50		50	8E		03	7B 004DA	EDIV	#3, (SP)+, R0, R0	
					50	D5 004DF	TSTL	R0	
					03	13 004E1	BEQL	86\$	
				00E0	31 004E3	BRW	101\$		
			04		03	C6 004E6	DIVL2	#3, KEY_COUNT	1125
			AE		01	CE 004EA	MNEGL	#1, KEY_NUM	1131
				00C1	31 004ED	BRW	97\$		
			7E	50	8F	9A 004F0	MOVZBL	#80, -(SP)	1133
00000000G		00			01	FB 004F4	CALLS	#1, FOR\$\$GET_VM	

0050	8F	00	10 04	AE A8 58 6E	10 10	50 AE D0 00 68 8F A6 03 02F3 52 F4 A6 EA A6 E4 01 52 52 CF 52 A8 A8 A8 0E	D0 D0 D0 2C B0 D0 14 31 D1 14 D1 14 D1 19 A3 C3 D6 D1 14 90 B0 90 95 13 91 12 D4 11 91 13 91 13 91 12 91 12 D0 11 D0 11 91 12 91 12 D0 11 D0 90 90 D5 13 88 90 C0 F2 11	004FB 004FF 00504 00508 0050F 00510 00515 00519 0051B 0051E 00525 00527 0052F 00531 00535 00537 0053C 00541 00543 0054A 0054C 00550 00555 0055A 0055C 0055E 00561 00563 00565 00567 0056A 0056C 0056F 00571 00574 00576 0057A 0057C 0057F 00581 00584 00586 00589 0058B 0058F 00591 00594 00596 00599 0059D 005A2 005A4 005A6 005AA 005AE 005B1 005B6	MOV MOV MOV MOV MOV MOV BGTR BRW CMPL BGTR CMPL BGTR CMPL BLSS SUBW3 SUBL3 INCL CMPL BGTR MOVB MOVW MOVB TSTB BEQL CMPB BNEQ CLRL BRB CMPB BEQL CMPB BEQL CMPB BNEQ CMPB BNEQ MOV BRB MOV BRB CMPB BNEQ CMPB BNEQ MOV BRB MOV MOVB TSTL BEQL BISB2 MOVB ADDL2 AOBLS5 BRB	R0, XAB_ADDR XAB_ADDR, 4(KEY_XAB) XAB_ADDR, KEY_XAB #0, (SP), #0, #80, (KEY_XAB) #19477, (KEY_XAB) 4(KEY_DEFN), R2 89\$ 141\$ R2, #32767 88\$ 8(KEY_DEFN), #32767 88\$ 8(KEY_DEFN), R2 88\$ #1, R2, 30(KEY_XAB) R2, 8(KEY_DEFN), R2 SIZE SIZE, #255 88\$ SIZE, 46(KEY_XAB) 30(KEY_XAB), 78(KEY_XAB) 46(KEY_XAB), 77(KEY_XAB) (KEY_DEFN) 90\$ (KEY_DEFN), #14 91\$ R0 95\$ (KEY_DEFN), #3 92\$ (KEY_DEFN), #7 94\$ (KEY_DEFN), #4 93\$ 46(KEY_XAB), #4 92\$ #4, R0 95\$ #2, R0 95\$ (KEY_DEFN), #8 101\$ 46(KEY_XAB), #4 94\$ #3, R0 95\$ #1, R0 R0, 19(KEY_XAB) 19(KEY_XAB), 76(KEY_XAB) KEY_NUM 96\$ #3, 18(KEY_XAB) KEY_NUM, 23(KEY_XAB) #12, KEY_DEFN KEY_COUNT, KEY_NUM, 98\$ 99\$	1134 1135 1141 1142 1149 1150 1151 1152 1156 1163 1165 1167 1169 1170 1173 1174 1175 1176 1177 1171 1183 1185 1189 1192 1193 1131
------	----	----	----------	----------------------	----------	--	--	---	--	---	--

02	00	60	FF35	31	005B8	98\$:	BRW	87\$		
000B	0010		A7	CF	005BB	99\$:	CASEL	96(R7), #0, #2		1204
			0010		005C0	100\$:	.WORD	103\$-100\$,-		
								103\$-100\$,-		
								102\$-100\$		
			30	DD	005C6	101\$:	PUSHL	#48		1214
			0557	31	005C8		BRW	214\$		
	FF	AB	40	8F	88	005CB	102\$:	BISB2	#64, -1(CCB)	1211
		50	55	D4	005D0	103\$:	CLRL	V_DEFAULT_SIZE		1230
			18	A7	D0	005D2	MOVL	24(R7), R0		1232
				3C	12	005D6	BNEQ	110\$		1235
			D2	AB	B5	005D8	TSTW	-46(CCB)		1243
				6D	12	005DB	BNEQ	115\$		
OE	FC	AB		03	E0	005DD	BBS	#3, -4(CCB), 106\$		1247
03	FD	AB		02	E1	005E2	BBC	#2, -3(CCB), 105\$		
		10		03F6	31	005E7	104\$:	BRW	188\$	
			61	AB	91	005EA	105\$:	CMPB	97(CCB), #16	1248
				F7	13	005EE	BEQL	104\$		
12	FD	AB		01	E1	005F0	106\$:	BBC	#1, -3(CCB), 108\$	1253
06	FD	AB		02	E1	005F5	BBC	#2, -3(CCB), 107\$		1255
		50	80	8F	9A	005FA	MOVZBL	#128, R0		
				0B	11	005FE	BRB	109\$		
		50	07FC	8F	3C	00600	107\$:	MOVZWL	#2044, R0	
				04	11	00605	BRB	109\$		
		50	85	8F	9A	00607	108\$:	MOVZBL	#133, R0	1253
	D2	AB		50	B0	0060B	109\$:	MOVW	R0, -46(CCB)	1252
		55		01	D0	0060F	MOVL	#1, V_DEFAULT_SIZE		1262
	00007FFF	BF		36	11	00612	BRB	115\$		1243
				50	D1	00614	110\$:	CMPB	R0, #32767	1265
				CA	1A	0061B	BGTRU	104\$		
05	FD	AB		01	E1	0061D	BBC	#1, -3(CCB), 111\$		1271
		51		04	D0	00622	MOVL	#4, R1		
				03	11	00625	BRB	112\$		
		51		01	D0	00627	111\$:	MOVL	#1, R1	
		51		50	C4	0062A	112\$:	MULL2	R0, R1	
05	FD	AB		03	E1	0062D	BBC	#3, -3(CCB), 113\$		1272
		50		02	D0	00632	MOVL	#2, R0		
				02	11	00635	BRB	114\$		
				50	D4	00637	113\$:	CLRL	R0	
52		51		50	C1	00639	114\$:	ADDL3	R0, R1, T	
	00007FFF	8F		52	D1	0063D	CMPB	T, #32767		1274
				A1	1A	00644	BGTRU	104\$		
	D2	AB		52	B0	00646	MOVW	T, -46(CCB)		1276
		56	FC	AB	9E	0064A	115\$:	MOVAB	-4(CCB), R6	1283
0C		66		0A	E0	0064E	BBS	#10, (R6), 116\$		
		10	61	AB	91	00652	CMPB	97(CCB), #16		1284
				06	13	00656	BEQL	116\$		
		20	61	AB	91	00658	CMPB	97(CCB), #32		1285
				05	12	0065C	BNEQ	117\$		
	7A	AB	D2	AB	B0	0065E	116\$:	MOVW	-46(CCB), 122(CCB)	1286
			28	A7	D5	00663	117\$:	TSTL	40(R7)	1296
				13	13	00666	BEQL	119\$		
		50	28	A7	D0	00668	MOVL	40(R7), R0		1299
				03	18	0066C	BGEQ	118\$		
		50		50	CE	0066E	MNEGL	R0, R0		
	54	AB		50	D0	00671	118\$:	MOVL	R0, 84(CCB)	
GC	BE	01		01	F0	00675	INSV	#1, #21, #1, @12(SP)		1300

00	BE	01	58	AB	03	30	A7	D5	0067B	119\$:	TSTL	44(R7)	1308
							16	13	0067E		BEQL	121\$	
			50			2C	A7	D0	00680		MOVL	44(R7), R0	1311
							03	18	00684		BGEQ	120\$	
			50				50	CE	00686		MNEGL	R0, R0	
		00010000	8F				50	D1	00689	120\$:	CMPL	R0, #65536	
							65	1E	00690		BGEQU	124\$	
			58	AB			50	B0	00692		MOVW	R0, 88(CCB)	1313
							A7	F0	00696	121\$:	INSV	48(R7), #3, #1, a0(SP)	1321
						40	A7	D5	0069D		TSTL	64(R7)	1329
							05	13	006A0		BEQL	122\$	
			E4	AB		40	A7	D0	006A2		MOVL	64(R7), -28(CCB)	
			7C	AB		E4	AB	D0	006A7	122\$:	MOVL	-28(CCB), 124(CCB)	1331
						48	A7	D0	006AC		MOVL	72(R7), R0	1341
							30	13	006B0		BEQL	123\$	1344
		0000FFFF	8F				50	D1	006B2		CMPL	R0, #65535	1347
							3C	1A	006B9		BGTRU	124\$	
		0080	CB				50	B0	006BB		MOVW	R0, 128(CCB)	1349
			50		01FF		CO	9E	006C0		MOVAB	511(R0), R0	1350
			50		00000200		8F	C7	006C5		DIVL3	#512, R0, R1	
			37	AB			51	90	006CD		MOVW	R1, 55(CCB)	
							CB	9E	006D1		MOVAB	130(CCB), R0	1351
					0082		AB	90	006D6		MOVW	55(CCB), (R0)	
					37		60	91	006DA		CMPL	(R0), #63	1352
							03	1B	006DD		BLEQU	123\$	
			60				3F	90	006DF		MOVW	#63, (R0)	1354
			50			24	A7	D0	006E2	123\$:	MOVL	36(R7), R0	1370
							14	13	006E6		BEQL	125\$	1373
		0000007F	8F				50	D1	006E8		CMPL	R0, #127	1376
							06	1A	006EF		BGTRU	124\$	
			36	AB			50	90	006F1		MOVW	R0, 54(CCB)	1377
							05	11	006F5		BRB	125\$	
							2D	D0	006F7	124\$:	PUSHL	#45	1380
							D426	31	006F9		BRW	214\$	
						44	A7	D5	006FC	125\$:	TSTL	68(R7)	1387
							0C	13	006FF		BEQL	126\$	
			DC	AB		44	A7	D0	00701		MOVL	68(R7), -36(CCB)	1390
							67	E9	00706		BLBC	(R7), 126\$	1392
			01	A6			10	88	00709		BISB2	#16, 1(R6)	
						54	A7	D5	0070D	126\$:	TSTL	84(R7)	1407
							17	13	00710		BEQL	127\$	
			2C	AE		C6	AB	32	00712		CVTWL	-58(CCB), LOG_UNIT	1414
			01	A9			04	88	00717		BISB2	#4, 1(R9)	1415
						2C	AE	9F	0071B		PUSHAB	LOG_UNIT	1416
							5B	DD	0071E		PUSHL	CCB	1417
						10	AE	DD	00720		PUSHL	16(SP)	1416
			54	B7			03	FB	00723		CALLS	#3, a84(R7)	
							29	11	00727		BRB	130\$	
							03	E1	00729	127\$:	BBC	#3, (R6), 128\$	1432
						08	AE	DD	0072D		PUSHL	8(SP)	1434
		00000000G	00				01	FB	00730		CALLS	#1, SYS\$OPEN	
							0A	11	00737		BRB	129\$	
						08	AE	DD	00739	128\$:	PUSHL	8(SP)	1436
		00000000G	00				01	FB	0073C		CALLS	#1, SYS\$CREATE	
			52				50	D0	00743	129\$:	MOVL	R0, OPEN_STATUS	
			0C				52	E9	00746		BLBC	OPEN_STATUS, 131\$	1442
							5B	DD	00749		PUSHL	CCB	

00000000G	00	01	FB	0074B	CALLS	#1, SYSSCONNECT	
	52	50	D0	00752	130\$:	RO, OPEN_STATUS	
		68	AB	D4	00755	131\$:	CLRL 104(CCB)
OD	OC	19	E1	00758		BBC	#25, @12(SP), 132\$
00010619	8F	4C	AB	D1	0075D		CMPL 76(CCB), #67097
			03	13	00765		BEQL 132\$
	66		08	88	00767		BISB2 #8, (R6)
OE	FE		00	E5	0076A	132\$:	BBCC #0, -2(CCB), 133\$
		14	BE	DD	0076F		PUSHL @20(SP)
	7E	20	BE	9A	00772		MOVZBL @32(SP), -(SP)
00000000G	00		02	FB	00776		CALLS #2, FOR\$FREE_VM
		0097	CB	95	0077D	133\$:	TSTB 151(CCB)
			0D	13	00781		BEQL 134\$
	14	24	BE	D0	00783		MOVL @36(SP), @20(SP)
	1C	0097	CB	90	00788		MOVB 151(CCB), @28(SP)
			11	11	0078E		BRB 135\$
		009F	CB	95	00790	134\$:	TSTB 159(CCB)
			08	13	00794		BEQL 135\$
	14	20	BE	D0	00796		MOVL @32(SP), @20(SP)
	1C	009F	CB	90	0079B		MOVB 159(CCB), @28(SP)
			52	E9	007A1	135\$:	BLBC OPEN_STATUS, 136\$
		00CB	31	007A4		BRW	150\$
00018292	50	4C	AB	D0	007A7	136\$:	MOVL 76(CCB), R0
	8F		50	D1	007AB		CMPL R0, #98962
			04	12	007B2		BNEQ 137\$
			1D	DD	007B4		PUSHL #29
			5B	11	007B6		BRB 142\$
000184C4	8F		50	D1	007B8	137\$:	CMPL R0, #99524
			04	12	007BF		BNEQ 138\$
			2A	DD	007C1		PUSHL #42
			4E	11	007C3		BRB 142\$
0001852C	8F		50	D1	007C5	138\$:	CMPL R0, #99628
			24	13	007CC		BEQL 139\$
000185F4	8F		50	D1	007CE		CMPL R0, #99828
			1B	13	007D5		BEQL 139\$
000186D4	8F		50	D1	007D7		CMPL R0, #100052
			12	13	007DE		BEQL 139\$
000186E4	8F		50	D1	007E0		CMPL R0, #100068
			09	13	007E7		BEQL 139\$
000186FC	8F		50	D1	007E9		CMPL R0, #100092
			04	12	007F0		BNEQ 140\$
			2B	DD	007F2	139\$:	PUSHL #43
			79	11	007F4		BRB 149\$
000185FC	8F		50	D1	007F6	140\$:	CMPL R0, #99836
			12	13	007FD		BEQL 141\$
00018624	8F		50	D1	007FF		CMPL R0, #99876
			09	13	00806		BEQL 141\$
0001868C	8F		50	D1	00808		CMPL R0, #100028
			04	12	0080F		BNEQ 143\$
			31	DD	00811	141\$:	PUSHL #49
			5A	11	00813	142\$:	BRB 149\$
0001C00A	8F		50	D1	00815	143\$:	CMPL R0, #114698
			4F	12	0081C		BNEQ 148\$
	53		50	D0	0081E		MOVL R0, OLD_STS
	52	50	AB	D0	00821		MOVL 80(CCB), OLD_STV
		08	AE	DD	00825		PUSHL 8(SP)
00000000G	00	01	FB	0082B	CALLS	#1, SYSPARSE	

			34		50	E9	0082F		BLBC	R0, 146\$		
		4C	AB		53	D0	00832		MOVL	OLD-STS, 76(CCB)		
		50	AB		52	D0	00836		MOVL	OLD-STV, 80(CCB)		
	26	0084	CB		05	E1	0083A		BBC	#5-132(CCB), 146\$		
				0080	CB	B5	00840		TSTW	128(CCB)		
	05		66		20	13	00844		BEQL	146\$		
			50		0A	E0	00846		BBS	#10, (R6), 144\$		
					04	D0	0084A		MOVL	#4, R0		
					02	11	0084D		BRB	145\$		
					50	D4	0084F	144\$:	CLRL	R0		
			51	D2	AB	3C	00851	145\$:	MOVZWL	-46(CCB), R1		
			50		51	C0	00855		ADDL2	R1, R0		
50	0080	CB	10		00	ED	00858		CMPZV	#0, #16, 128(CCB), R0		
					05	1E	0085F		BGEQU	146\$		
			50		25	D0	00861		MOVL	#37, R0		
					03	11	00864		BRB	147\$		
			50		1E	D0	00866	146\$:	MOVL	#30, R0		
					50	DD	00869	147\$:	PUSHL	R0		
					02	11	0086B		BRB	149\$		
					1E	DD	0086D	148\$:	PUSHL	#30		
	03		66		02B0	31	0086F	149\$:	BRW	214\$		
					03	E0	00872	150\$:	BBS	#3, (R6), 151\$		1576
					01D1	31	00876		BRW	199\$		
				4C	A7	D5	00879	151\$:	TSTL	76(R7)		1587
					21	13	0087C		BEQL	157\$		
	02		01	4C	A7	CF	0087E		CASEL	76(R7), #1, #2		1594
	0011		000C		0008		00883	152\$:	.WORD	153\$-152\$,-		
										154\$-152\$,-		
										155\$-152\$		
					4A	11	00889		BRB	163\$		1601
					50	D4	0088B	153\$:	CLRL	T		1594
					08	11	0088D		BRB	156\$		
			50		10	D0	0088F	154\$:	MOVL	#16, T		
					03	11	00892		BRB	156\$		
			50		20	D0	00894	155\$:	MOVL	#32, T		
50	61	AB	08		00	ED	00897	156\$:	CMPZV	#0, #8, 97(CCB), T		1605
					14	12	0089D		BNEQ	159\$		
					66	B5	0089F	157\$:	TSTW	(R6)		1614
					06	18	008A1		BGEQ	158\$		
			20	61	AB	91	008A3		CMPB	97(CCB), #32		
					0A	12	008A7		BNEQ	159\$		
	09		66		04	E1	008A9	158\$:	BBC	#4, (R6), 160\$		
			20	61	AB	91	008AD		CMPB	97(CCB), #32		1615
					03	12	008B1		BNEQ	160\$		
			54	61	026A	31	008B3	159\$:	BRW	213\$		
					AB	9E	008B6	160\$:	MOVAB	97(CCB), R4		1623
					64	95	008BA		TSTB	(R4)		
					04	13	008BC		BEQL	161\$		
		01	A9		08	88	008BE		BISB2	#8, 1(R9)		
	06		00	50	A7	CF	008C2	161\$:	CASEL	80(R7), #0, #6		1635
0055	0049		0043		0011		008C7	162\$:	.WORD	164\$-162\$,-		
	006D		0067		0061		008CF			167\$-162\$,-		
										168\$-162\$,-		
										169\$-162\$,-		
										170\$-162\$,-		
										171\$-162\$,-		
										172\$-162\$		

			FCEE	31	008D5	163\$:	BRW	101\$	1693	
	01	A6	0C	8A	008D8	164\$:	BICB2	#12, 1(R6)	1641	
		U1	63	AB	91	008DC	CMPB	99(CCB), #1	1643	
				06	12	008E0	BNEQ	165\$		
	01	A6		04	88	008E2	BISB2	#4, 1(R6)	1645	
				57	11	008E6	BRB	175\$		
06		66		04	E1	008E8	165\$:	BBC	#4, (R6), 166\$	1648
4F		69		08	E0	008EC	BBS	#11, (R9), 175\$		
				48	11	008F0	BRB	174\$	1650	
49		69		08	E0	008F2	166\$:	BBS	#11, (R9), 175\$	1651
45		66		09	E1	008F6	BBC	#9, (R6), 175\$		
41		66		04	E0	008FA	BBS	#4, (R6), 175\$	1652	
		02	63	AB	91	008FE	CMPB	99(CCB), #2		
				38	12	00902	BNEQ	175\$		
	01	A6		08	88	00904	BISB2	#8, 1(R6)	1654	
				35	11	00908	BRB	175\$	1635	
		01	63	AB	91	0090A	167\$:	CMPB	99(CCB), #1	1660
				28	11	0090E	BRB	173\$		
		02	63	AB	91	00910	168\$:	CMPB	99(CCB), #2	1664
				29	13	00914	BEQL	175\$		
		03	63	AB	91	00916	CMPB	99(CCB), #3		
				1C	11	0091A	BRB	173\$		
		02	63	AB	91	0091C	169\$:	CMPB	99(CCB), #2	1670
				18	12	00920	BNEQ	174\$		
19		69		08	E1	00922	BBC	#11, (R9), 175\$		
				12	11	00926	BRB	174\$	1672	
		04	63	AB	91	00928	170\$:	CMPB	99(CCB), #4	1676
				0A	11	0092C	BRB	173\$		
		06	63	AB	91	0092E	171\$:	CMPB	99(CCB), #6	1682
				04	11	00932	BRB	173\$		
		05	63	AB	91	00934	172\$:	CMPB	99(CCB), #5	1688
				05	13	00938	173\$:	BEQL	175\$	
				2C	DD	0093A	174\$:	PUSHL	#44	1690
			01E3	31	0093C	BRW	214\$			
			E4	AB	D5	0093F	175\$:	TSTL	-28(CCB)	1700
				07	12	00942	BNEQ	176\$		
	E4	AB	7C	AB	D0	00944	MOVL	124(CCB), -28(CCB)	1702	
				17	11	00949	BRB	178\$		
			7C	AB	D5	0094B	176\$:	TSTL	124(CCB)	1705
				12	13	0094E	BEQL	178\$		
		50	E4	AB	D0	00950	MOVL	-28(CCB), R0		
	7C	AB		50	D1	00954	CMPL	R0, 124(CCB)		
				04	15	00958	BLEQ	177\$		
		50	7C	AB	D0	0095A	MOVL	124(CCB), R0		
	E4	AB		50	D0	0095E	177\$:	MOVL	R0, -28(CCB)	
11	0087	CB		04	E0	00962	178\$:	BBS	#4, 135(CCB), 179\$	1724
0B	0084	CB		02	E0	00968	BBS	#2, 132(CCB), 179\$	1725	
			7A	AB	B5	0096E	TSTW	122(CCB)	1727	
				06	12	00971	BNEQ	179\$		
	7A	AB	0080	CB	B0	00973	MOVW	128(CCB), 122(CCB)	1729	
		10		55	E8	00979	179\$:	BLBS	V DEFAULT SIZE, 181\$	1731
05		66		0A	E0	0097C	BBS	#10, (R6), 180\$		
		10		64	91	00980	CMPB	(R4), #16	1732	
				07	12	00983	BNEQ	181\$		
	7A	AB	D2	AB	B1	00985	180\$:	CMPL	-46(CCB), 122(CCB)	1735
				54	12	0098A	BNEQ	188\$		
		52	D2	AB	9E	0098C	181\$:	MOVAB	-46(CCB), R2	1740

05	66	0A	E0	00990	BBS	#10, (R6), 182\$	1737
	10	64	91	00994	CMPB	(R4), #16	1738
	62	7A	AB	B0 00999	BNEQ	183\$	1740
			1A	11 0099D	MOVW	122(CCB), (R2)	
	50		62	3C 0099F	BRB	186\$	1742
	50	7A	AB	B1 009A2	MOVZWL	(R2), R0	
			04	1B 009A6	CMPW	122(CCB), R0	
	50	7A	AB	3C 009A8	BLEQU	184\$	
	50	D2	AD	B1 009AC	MOVZWL	122(CCB), R0	
			04	1B 009B0	CMPW	XAB_BLOCK+10, R0	
	50	D2	AD	3C 009B2	BLEQU	185\$	
	62		50	B0 009B6	MOVZWL	XAB_BLOCK+10, R0	
	20		64	91 009B9	MOVW	R0, (R2)	
			29	12 009BC	CMPB	(R4), #32	1744
25	66		0A	E0 009BE	BNEQ	190\$	
	53	7A	AB	3C 009C2	BBS	#10, (R6), 190\$	1751
			10	12 009C6	MOVZWL	122(CCB), R3	
	1C		55	E9 009C8	BNEQ	187\$	
	50	0082	CB	9A 009CB	BLBC	V_DEFAULT_SIZE, 190\$	1754
62	50	0200	8F	A5 009D0	MOVZBL	130(CCB), R0	1756
			0F	11 009D6	MULW3	#512, R0, (R2)	
	09		55	E8 009D8	BRB	190\$	1751
	53		62	B1 009DB	BLBS	V_DEFAULT_SIZE, 189\$	1764
			04	1B 009DE	CMPW	(R2), R3	1765
			25	DD 009E0	BLEQU	189\$	
			63	11 009E2	PUSHL	#37	1767
	62		53	B0 009E4	BRB	198\$	
	20		64	91 009E7	MOVW	R3, (R2)	1769
			62	12 009EA	CMPB	(R4), #32	1780
	50	18	AE	D0 009EC	BNEQ	200\$	
			06	13 009F0	MOVL	24(SP), KEY_DEFN	1793
	53	02	A0	32 009F2	BEQL	191\$	1795
			02	11 009F6	CVTWL	2(KEY_DEFN), KEY_COUNT	
			53	D4 009F8	BRB	192\$	
	52		01	D0 009FA	CLRL	KEY_COUNT	
	58	CC	AD	D0 009FD	MOVL	#1, XAB_STATUS	1799
			58	D5 00A01	MOVL	XAB_BLOCK+4, KEY_XAB	1800
			3D	13 00A03	TSTL	KEY_XAB	1802
			53	D5 00A05	BEQL	197\$	
			39	15 00A07	TSTL	KEY_COUNT	
	15		68	91 00A09	BLEQ	197\$	
			F3	12 00A0C	CMPB	(KEY_XAB), #21	1805
4E	A8	1E	A8	B1 00A0E	BNEQ	193\$	
			07	12 00A13	CMPW	30(KEY_XAB), 78(KEY_XAB)	1809
4D	A8	2E	A8	91 00A15	BNEQ	194\$	
			03	13 00A1A	CMPB	46(KEY_XAB), 77(KEY_XAB)	1810
	52		31	D0 00A1C	BEQL	195\$	
13	A8	4C	A8	91 00A1F	MOVL	#49, XAB_STATUS	1812
			03	13 00A24	CMPB	76(KEY_XAB), 19(KEY_XAB)	1814
	52		31	D0 00A26	BEQL	196\$	
	54	04	A8	D0 00A29	MOVL	#49, XAB_STATUS	1816
			58	DD 00A2D	MOVL	4(KEY_XAB), NEXT	1823
	7E	50	8F	9A 00A2F	PUSHL	KEY_XAB	1824
00000000G	00		02	FB 00A33	MOVZBL	#80, -(SP)	
	58		54	D0 00A3A	CALLS	#2, FOR\$\$FREE_VM	
	53		03	C2 00A3D	MOVL	NEXT, KEY_XAB	1825
					SUBL2	#3, KEY_COUNT	1827

			BF 11 00A40	BRB	193\$	1802
	09		52 E8 00A42 197\$:	BLBS	XAB STATUS, 200\$	1839
			52 DD 00A45	PUSHL	XAB STATUS	1841
			00D8 31 00A47 198\$:	BRW	214\$	
	01	A6	20 8A 00A4A 199\$:	BICB2	#32, 1(R6)	1856
		28	55 E9 00A4E 200\$:	BLBC	V DEFAULT SIZE, 202\$	1864
22	0087	CB	04 E0 00A51	BBS	#4, 135(CCB), 202\$	1866
1C	0084	CB	02 E0 00A57	BBS	#2, 132(CCB), 202\$	1867
			0080 CB B5 00A5D	TSTW	128(CCB)	1868
			16 13 00A61	BEQL	202\$	
		50	0080 CB B0 00A63	MOVW	128(CCB), NEW RECL	1873
03		66	0B E1 00A68	BBC	#11, (R6), 20T\$	1874
		50	04 A2 00A6C	SUBW2	#4, NEW RECL	1876
	D2	AB	50 B1 00A6F 201\$:	CMPW	NEW RECL, -46(CCB)	1877
			04 1E 00A73	BGEQU	202\$	
	D2	AB	50 B0 00A75	MOVW	NEW RECL, -46(CCB)	1879
		05	00CA CB E9 00A79 202\$:	BLBC	202(CCB), 203\$	1888
	00	BE	28 AE 90 00A7E	MOVB	ORIG RAT, @0(SP)	1890
		06	55 E8 00A83 203\$:	BLBS	V DEFAULT SIZE, 204\$	1897
		50	D2 AB 3C 00A86	MOVZWL	-46(CCB), -R0	
			0E 11 00A8A	BRB	206\$	
		06	00 BE E9 00A8C 204\$:	BLBC	@0(SP), 205\$	1898
		50	51 8F 9A 00A90	MOVZBL	#81, R0	
			04 11 00A94	BRB	206\$	
		50	50 8F 9A 00A96 205\$:	MOVZBL	#80, R0	
	D4	AB	50 B0 00A9A 206\$:	MOVW	R0, -44(CCB)	1897
		04	00 BE E9 00A9E	BLBC	@0(SP), 207\$	1905
		69	80 8F 88 00AA2	BISB2	#128, (R9)	1907
04	00	BE	01 E1 00AA6 207\$:	BBC	#1, @0(SP), 208\$	1908
		69	40 8F 88 00AAB	BISB2	#64, (R9)	1910
04	00	BE	02 E1 00AAF 208\$:	BBC	#2, @0(SP), 209\$	1911
	01	A9	01 88 00AB4	BISB2	#1, 1(R9)	1913
		7E	D2 AB 3C 00AB8 209\$:	MOVZWL	-46(CCB), -(SP)	1920
	00000000G	00	01 FB 00ABC	CALLS	#1, FOR\$\$GET_VM	
	EC	AB	50 D0 00AC3	MOVL	R0, -20(CCB)	
		7E	1C BE 9A 00AC7	MOVZBL	@28(SP), -(SP)	1935
	00000000G	00	01 FB 00ACB	CALLS	#1, FOR\$\$GET_VM	
		57	50 D0 00AD2	MOVL	R0, T	
		50	1C BE 9A 00AD5	MOVZBL	@28(SP), R0	1936
		58	14 BE D0 00AD9	MOVL	@20(SP), R8	
67		68	50 28 00ADD	MOV C3	R0, (R8), (T)	
	14	BE	57 D0 00AE1	MOVL	T, @20(SP)	1937
	24	BE	57 D0 00AE5	MOVL	T, @36(SP)	1938
	20	BE	57 D0 00AE9	MOVL	T, @32(SP)	1939
	009F	CB	1C BE 90 00AED	MOVB	@28(SP), 159(CCB)	1940
	FE	AB	01 88 00AF3	BISB2	#1, -2(CCB)	1941
		50	61 AB 9A 00AF7	MOVZBL	97(CCB), R0	1948
			06 12 00AFB	BNEQ	210\$	1951
	C4	AB	01 90 00AFD	MOVB	#1, -60(CCB)	1952
		10	27 11 00B01	BRB	215\$	
			50 91 00B03 210\$:	CMPB	R0, #16	1954
			06 12 00B06	BNEQ	211\$	
	C4	AB	02 90 00B08	MOVB	#2, -60(CCB)	1955
		20	1C 11 00B0C	BRB	215\$	
			50 91 00B0E 211\$:	CMPB	R0, #32	1957
03		66	0D 12 00B11	BNEQ	213\$	
			0B E1 00B13	BBC	#11, (R6), 212\$	1960

			FE20	31	00B17		BRW	174\$		
	C4	AB	03	90	00B1A	212\$:	MOVW	#3, -60(CCB)	1962	
			0A	11	00B1E		BRB	215\$	1948	
			33	DD	00B20	213\$:	PUSHL	#51	1966	
	00000000G	00	01	FB	00B22	214\$:	CALLS	#1, FOR\$\$SIGNAL_STO		
				04	00B29		RET			
	24	AB	EC	AB	D0	00B2A	215\$:	MOVL	-20(CCB), 36(CCB)	1973
	20	AB	D2	AB	B0	00B2F		MOVW	-46(CCB), 32(CCB)	1974
	9C	AB	EC	AB	D0	00B34		MOVL	-20(CCB), -100(CCB)	1975
4C		69		0B	E0	00B39		BBS	#11, (R9), 217\$	1982
48		66		04	E0	00B3D		BBS	#4, (R6), 217\$	1983
44		66		0A	E0	00B41		BBS	#10, (R6), 217\$	1984
		3F	00CA	CB	E8	00B45		BLBS	202(CCB), 217\$	1985
3A	0C	BE		06	E0	00B4A		BBS	#6, @12(SP), 217\$	1986
34	0087	CB		04	E1	00B4F		BBC	#4, 135(CCB), 217\$	1991
		7E	0190	8F	3C	00B55		MOVZWL	#400, -(SP)	2003
	00000000G	00		01	FB	00B5A		CALLS	#1, FOR\$\$GET_VM	
	C8	AB		50	D0	00B61		MOVL	RCE, -56(CCB)	2010
	CC	AB		50	D0	00B65		MOVL	RCE, -52(CCB)	2011
		51	017C	C0	9E	00B69		MOVAB	380(R0), OLD_RCE	2012
		52		14	D0	00B6E		MOVL	#20, 1	2013
		61		50	D0	00B71	216\$:	MOVL	RCE, (OLD_RCE)	2015
	04	A0		51	D0	00B74		MOVL	OLD_RCE, 4(RCE)	2016
			08	A0	D4	00B78		CLRL	8(RCE)	2017
		51		80	7E	00B7B		MOVAQ	(RCE)+, OLD_RCE	2018
		50		0C	C0	00B7E		ADDL2	#12, RCE	2019
				52	D7	00B81		DECL	1	2013
			EC	12	00B83		BNEQ	216\$		
	01	A9		20	88	00B85		BISB2	#32, 1(R9)	2022
	D8	AB		02	90	00B89	217\$:	MOVW	#2, -40(CCB)	2029
		66		01	88	00B8D		BISB2	#1, (R6)	2030
		07	00000000G	00	E8	00B90		BLBS	FOR\$\$L_XIT_LOCK, 218\$	2036
	00000000G	00		00	FB	00B97		CALLS	#0, FOR\$\$DECL_EXITH	
				04	00B9E	218\$:	RET			2039

; Routine Size: 2975 bytes, Routine Base: _FOR\$CODE + 0090

: 1999 2040 1
: 2000 2041 1 END
: 2001 2042 1
: 2002 2043 0 ELUDOM

! End of FOR\$OPEN_DEFLT module

PSECT SUMMARY

Name	Bytes	Attributes
_FOR\$CODE	3119	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

FOR\$OPEN_DEFLT FORTRAN default open
1-098

I 13
16-Sep-1984 00:37:00
14-Sep-1984 12:32:16

VAX-11 Bliss-32 V4.0-742
[FORRTL.SRC]FOROPENDE.B32;1

Page 62
(25)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	127	1	581	00:01.1
-\$255\$DUA28:[FORRTL.OBJ]FORLIB.L32;1	711	283	39	52	00:00.6
-\$255\$DUA28:[FORRTL.OBJ]RTLLIB.L32;1	36	0	0	8	00:00.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:FOROPENDE/OBJ=OBJ\$:FOROPENDE MSRC\$:FOROPENDE/UPDATE=(ENH\$:FOROPENDE)

: Size: 3012 code + 107 data bytes
: Run Time: 01:26.6
: Elapsed Time: 03:27.2
: Lines/CPU Min: 1414
: Lexemes/CPU-Min: 15974
: Memory Used: 1167 pages
: Compilation Complete

0182 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY